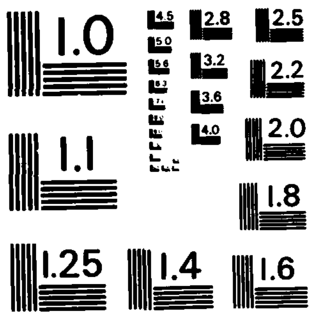
AD-A132 948 RADOME BORESIGHT ERROR ASSESSMENT AND SYSTEMS EVALUATION TEST CHAMBER IMP..(U) ALABAMA UNIV IN HUNTSVILLE SCHOOL OF ENGINEERING P A TILLEY ET AL. 1/1

UNCLASSIFIED APR 83 DRSMI/RD-CR-83-16 DAAH01-82-D-A008 F/G 17/9 NL

END
DATE
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-E950 426

(11)

AD-A132 948

TECHNICAL REPORT RD-CR-83-16

RANDOME BORESIGHT ERROR ASSESSMENT AND SYSTEMS EVALUATION TEST CHAMBER IMPROVEMENTS

Patrick A. Tilley, Roger G. Gean, and
Timothy A. Palmer
The University of Alabama in Huntsville
School of Engineering
Huntsville, Alabama 35899

April 1983

Prepared for
System Simulation and Development Directorate
US Army Missile Laboratory

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Approved for public release; distribution unlimited.

SMI FORM 1021, 1 JUL 79 PREVIOUS EDITION IS OBSOLETE

DISPOSITION INSTRUCTIONS

DISCLAIMER

TRADE NAMES

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RD-CR-83-16	2. GOVT ACCESSION NO. AD-A132948	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RADOME BORESIGHT ERROR ASSESSMENT AND SYSTEMS EVALUATION TEST CHAMBER IMPROVEMENTS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 10/8/82 - 12/31/82
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Patrick Tilley, Roger Gean and Timothy Palmer		8. CONTRACT OR GRANT NUMBER(s) DAAH01-82-D-A008 Delivery Order #0006
9. PERFORMING ORGANIZATION NAME AND ADDRESS The University of Alabama in Huntsville School of Engineering Huntsville, AL 35899		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander US Army Missile Command ATTN: DRSMI-RPT Redstone Arsenal, AL 35898		12. REPORT DATE April 1983
		13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Commander US Army Missile Command ATTN: DRSMI-RD Redstone Arsenal, AL 35898		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distritution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Radome, Boresight Error, 3-D Mapping, Vector Plots, Statistical Analysis, System Evaluation Test Chamber, Radome Positioner, Radome Measurements Receiver System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Accomplishments in radome testing and analysis were continued through performance of the task documented by this report. Software improvements for the Radome Positioner and the Radome Measurements Receiver System are presented. The radome testing and post-processing of data are described. Testing and processing difficulties are also discussed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PREFACE

This technical report was prepared by R. G. Gean, P. A. Tilley, and T. A. Palmer, Electrical and Computer Engineering Department, School of Engineering, The University of Alabama in Huntsville. The purpose of this report is to provide documentation of the technical work performed and of results obtained under delivery order number 0006, contract number DAAH01-82-D-A008; Dr. N. A. Kheir, Principal Investigator. Dr. M. M. Hallum, III, Chief, System Evaluation Branch, Army Missile Laboratory, U.S. Army Missile Command, was technical monitor. Mr. Ernst Evers-Euterneck of the Systems Evaluation Branch provided technical coordination.

The authors wish to acknowledge the valuable discussions and assistance provided throughout the task by L. Ragland of the Systems Evaluation Branch and C. Adams of the Aeroballistic Analysis Branch.

The technical viewpoints, opinions, and conclusions herein are those of the authors and do not imply policies or positions of the U.S. Army Missile Command.

TABLE OF CONTENTS

	Page
PREFACE	i
TABLE OF CONTENTS	ii
1.0 INTRODUCTION	1
2.0 SYSTEM EVALUATION TEST CHAMBER (SETC) PREPARATION	2
2.1 Introduction	2
2.2 Phase Offset	2
2.3 Software Improvements	2
2.3.1 Radome Positioner Improvements	2
2.3.2 Radome Measurements Receiver System Improvements	4
3.0 SETC TESTING	6
3.1 Introduction	6
3.2 Group 1 - Acceptable Radomes	6
3.3 Group 2 - Obstructed Radome	6
3.4 Group 3 - No Radome	6
4.0 DATA PROCESSING	7
4.1 Introduction	7
4.2 Removing Offset	7
4.3 Plotting Programs	7
4.4 Statistical Analysis	7
5.0 TESTING AND PROCESSING DIFFICULTIES	8
5.1 Introduction	8
5.2 Equipment Problems	8
5.3 Phase Coordinate System	8
6.0 CONCLUSIONS AND RECOMMENDATIONS	10
REFERENCES	11

APPENDICES

A. Boresight Error Plots
B. Radome Positioner Software Listing
C. Radome Measurements Receiver System Software Listing

1.0 INTRODUCTION

The purpose of this report is to review the work performed and the results obtained since completion of the previous delivery order reported in [1]. The goal of this work was to provide a statistical evaluation of IHAWK radomes. The areas of work performed include System Evaluation Test Chamber (SETC) preparations, radome measurements, and data processing.

Section 2 of this report will present the hardware and software changes incorporated to improve the accuracy of data. Changes in test methods, the Radome Positioner software, and the Radome Measurements System software resulted in improved accuracy by improving the acquisition process. The actual tests performed and the results of those tests are found in Section 3 while Section 4 provides a description of the processing means used to analyze and to plot data. Section 5 presents an outline of the major problems and difficulties encountered during performance of the task. Conclusions and recommendations for possible future improvements are presented in Section 6.

2.0 SYSTEM EVALUATION TEST CHAMBER (SETC) PREPARATION

2.1 Introduction

SETC preparations continued throughout this task and included reconfiguring the hardware for each entrance to the SETC. This was essential since several groups share the SETC and each requires a different configuration.

A Modification to the test configuration was made by using an offset transmitter horn configuration to induce a known error. The error must stay sufficiently large for the network analyzer to register a stable phase measurement. Another SETC modification consisted of new software written in Motorola 6800 assembly code and Fortran for the Radome Positioner and Radome Measurements Receiver System. This software provides improved operating systems which produce more usable, more complete data sets.

2.2 Phase Offset

Phase offsets measured by the network analyzer provide the signs of error when plotting data. For radomes with very small boresight errors the network analyzer could not provide a stable phase measurement. Thus, a need existed to find a method to accurately test radomes with the available equipment.

Studying the problem led to offsetting the transmitting horn by a known amount. The receiving antenna was boresighted on this offset horn. After boresighting, the transmitting horn was returned to its original center position and scans were executed. Using this method, an inherent error (the amount of offset), which is of a magnitude large enough for the network analyzer to operate accurately, exists. This offset is later removed during data processing to reveal the true magnitude and phase of the boresight error. The arrangement of the transmitting horn is illustrated in Figure 1.

2.3 Software Improvements

Software improvements were made to both the radome positioner operating system and the radome measurements receiver system. Improvements in the positioner are discussed in Section 2.3.1, and those for the measurements receiver are discussed in Section 2.3.2.

2.3.1 Radome Positioner Improvements

A procedure has been developed for easy modification of the Radome Positioner operating system. Working from previous documentation, source files have been reconstructed and stored on master source file diskettes with a Tektronix 8002a microprocessor development system. Using these source files, the positioner program can be modified and recompiled as needed. Subsequently, PROMs can be programmed with modified programs and inserted into the M6800 based microcomputer which controls the positioner.

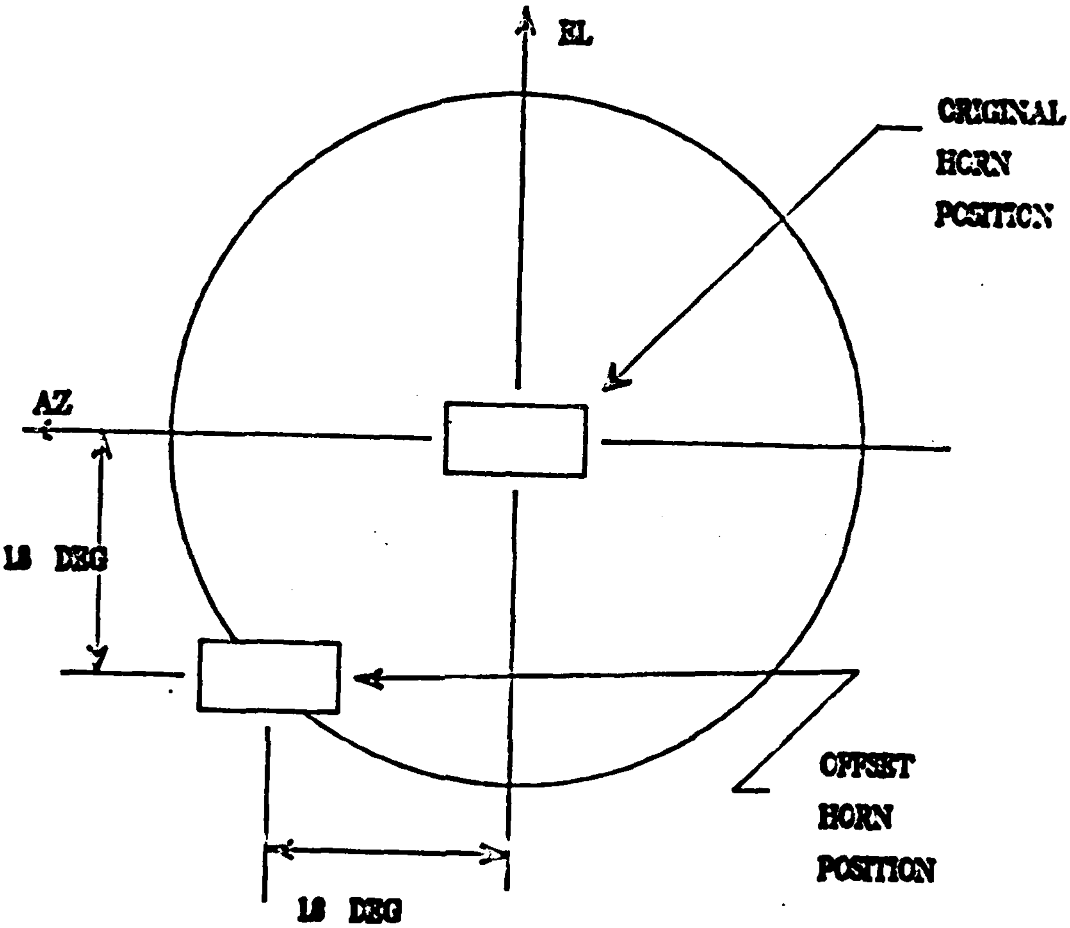


FIGURE 1 TRANSMITTING HORN ARRANGEMENT

Presently, all modifications are implemented using the Tektronix 8002a. However, the source files could be transferred to any other 6800 Development System.

Using the method described above, a new scan pattern has been implemented on the positioner. The previous positioner patterns were not adequate for thorough analysis of radome boresight error and antenna pattern measurements. This inadequacy existed because the area covered by the scan was not sufficient to yield the full scope of information about the antenna or radome. A careful analysis of the type of data needed revealed that the scan pattern illustrated in Figure 2 was the most useful pattern. The two most pertinent reasons were that (1) the shape of the radome suggested a circular window and (2) the method of data analysis intimated a raster-type scan.

The new pattern covers all data points necessary for current boresight error analysis and antenna pattern measurements. Execution of the pattern (number 3) requires the following key-in sequence:

```
PROG 3
  32.0
   1.0
```

where 32.0° is the radius of the circular scan region and 1.0 is the number of degrees between each azimuth scan. The revised operating system is presented in Appendix B.

2.3.2 Radome Measurements Receiver System Improvements

A procedure has been developed for modification of the Radome Measurement Receiver System operating system. Previously, it was believed that all calculations associated with radome boresight error and antenna pattern measurements could be done during "real-time" operation. However, it was determined that the receiver could not sample each point, perform all of the required calculation, and store the results before the radome reached the next sample point for the density of data desired. All of the calculation routines have therefore been removed from the operating system. This resulted in a more than 60% decrease in the total size of the object code file. In addition to this reduction, the FORTRAN main routine was decreased in size causing another 10% reduction of the total operating system size.

Previously, it was necessary for the operator to interact on a machine level in order to select a sample size. This interaction has been eliminated by enabling the FORTRAN main routine to pass the variable to the assembly routine, CNTL. Modifications have been made to CNTL to enable the receiver to sample on integer degrees within ± .043945°. This is important for precise calculations of boresight error slopes. The revised software is presented in Appendix C.

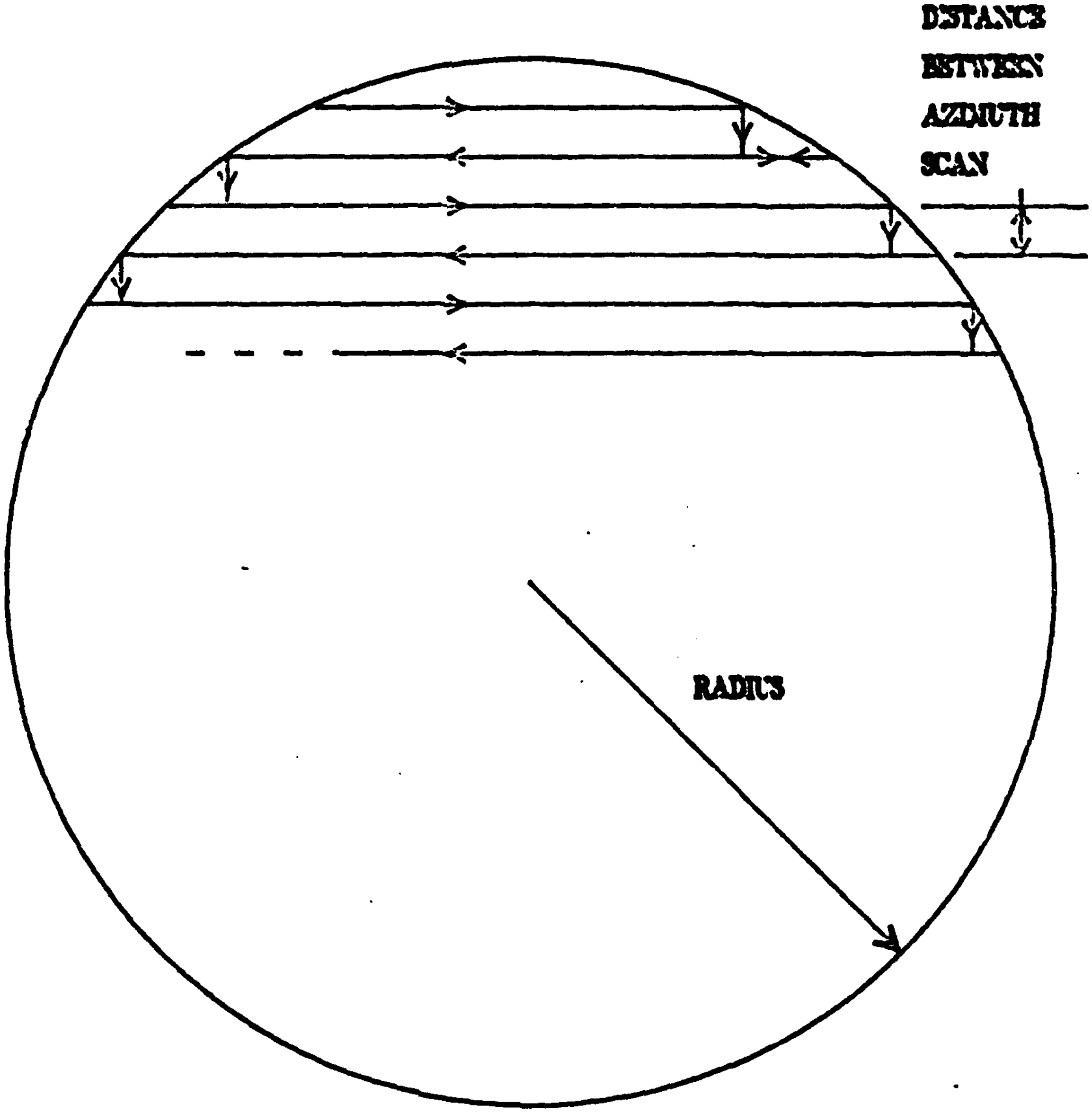


FIGURE 2 RASTER SCAN PATTERN

3.0 SETC TESTING

3.1 Introduction

The majority of tests covered by this report are tests which have been repeated using the induced offset described in Section 2.2. The tests fall into three groups: (1) accceptable radomes, (2) obstructed radomes, and (3) no radome.

3.2 GROUP 1 - Acceptable Radomes

This group of radomes include those which have passed final quality assurance inspections. Twelve IHAWK radomes in this category were tested. These tests were repeats of prior tests which did not use the induced offset test method. Vector plots obtained after removal of the known bias show uniformly distributed errors. The largest boresight errors are slightly in excess of one-half degree. The areas of least boresight error occur along the azimuth and elevation axes. Examples of these plots are presented in Figures A-1 through A-5 in Appendix A.

3.3 GROUP 2 - Obstructed Radomes

In an effort to better understand the offset horn configuration, an IHAWK radome was tested using a plexiglass obstruction. The purpose of this test was to observe the effect of the offset method on the large errors caused by placing an obstruction in the radome. Plots show that all errors not affected by the blockage are pointing toward the center of the radome. However, azimuth errors affected by blockage are pointing away from the center. The reason for this deviation may be a change in the polarization of the signal as it passes through the obstruction. Plots of this data are also included in Appendix A in Figures A-6 through A-10.

3.4 GROUP 3 - No Radome

Two tests were performed with no radome attached to the gimbal. The first test was run using the standard horn configurations. The purpose of this test was to check the testing equipment and the anechoic chamber; also, this test served as a comparison for the second test with no radome. The second test was made without a radome but with the horn positioned using the offset bias. This test was then compared with the previous test to detect any inconsistencies between the two configurations. The results of both tests were satisfactory with each showing only negligible errors.

4.0 DATA PROCESSING

4.1 Introduction

This section reviews the methods used to provide analysis of the data obtained during radome tests. The plotting programs and the statistical analysis program are discussed. Additionally, the method used to remove the induced bias are explained.

4.2 Removing Offset

With the configuration illustrated in Figure 1 and the dimensions of the anechoic chamber, there exists an offset of approximately 1.8° in both azimuth and elevation. Since exact positioning of the offset horn is unrealistic, the exact position can only be said to be within $\pm 0.1°$ of the desired value. The error, looking through the center of the radome, should be zero, and the overall average should also be zero. Thus, the average of all errors can be subtracted from each data point to compensate for small errors in alignment. Since the offset error is simply added to the true error, the total average of all errors is subtracted from each data point to compensate for both small errors in alignment and the $\pm$ 0.1° unknown offset error.

The average value is subtracted from each point of raw data with the result being stored in a new file containing the true boresight error for each set of azimuth and elevation angles. Date can then be retrieved by one or more of the plotting programs described in the following section. Plots of the data reflect the true boresight error values.

4.3 Plotting Programs

The two types of plots produced during this task were boresight error vector plots and boresight error three-dimensional plots. Minor adjustments to the existing vector plotting program allowed this program to access the file which contained data with the offset removed. These vector plots display azimuth and elevation errors in the form of a pointing vector emanating from a footpoint which is placed at integer values of azimuth and elevation angles.

Also utilized were two variations of the three-dimensional plotting routines. The first variation was the CARPET program which is normalized and produces a three-dimensional plot with all points connected by lines. The second was the program, CARPET 2, which is the same as CARPET except that it removes the lines which should be hidden, producing an easily readable plot. Examples of these plots are included in Appendix A, and a more detailed description of the actual plotting programs can be found in [1].

4.4 Statistical Analysis

The same methods of statistical analysis described in the report "Radome Boresight Assessment" [1] were utilized during this task. Data collected in the radome tests has been stored to facilitate rapid statistical analysis.

5.0 TESTING AND PROCESSING DIFFICULTIES

5.1 Introduction

Fewer difficulties arose during this task than during previous tasks. However, the following equipment and phase coordinate problems are considered significant.

5.2 Equipment Problems

At the beginning of the testing period, two radomes were tested and analyzed with the results being acceptable. Based on these results, the remaining tests were conducted prior to plotting and analyzing the data. This was required due to time limitations in the SETC. After testing was completed, discrepancies in phase were discovered in the last nine sets of data. This problem was traced to a phase line being grounded. After this problem was corrected, it was necessary to repeat the tests which yielded unacceptable data. Repeated tests provided acceptable results.

Another topic for discussion in this section is the computer system available for radome analysis. Although there has been a significant decrease in down time, terminal access remains a problem. Inavailability of terminals connected to the system has caused several delays.

5.3 Phase Coordinate System

Processing statistical evaluations of several sets of raw data unexpectedly revealed overall boresight averages with a negative azimuth value and with a positive elevation value. Since this was not expected while using the offset method described in Section 2.2, investigations to find the cause were conducted.

The problem was eventually solved by showing that the test equipment is arranged in a manner which logically reverses the apparent error during the test. For an observer positioned behind the antenna and looking in the direction of R_a (see Figure 3), a positive pitch (elevation) error indicates that the actual target location is above the R_a axis, as expected. However, a positive yaw (azimuth) error indicates that the actual target location is to the left of R_a in the yaw plane [2].

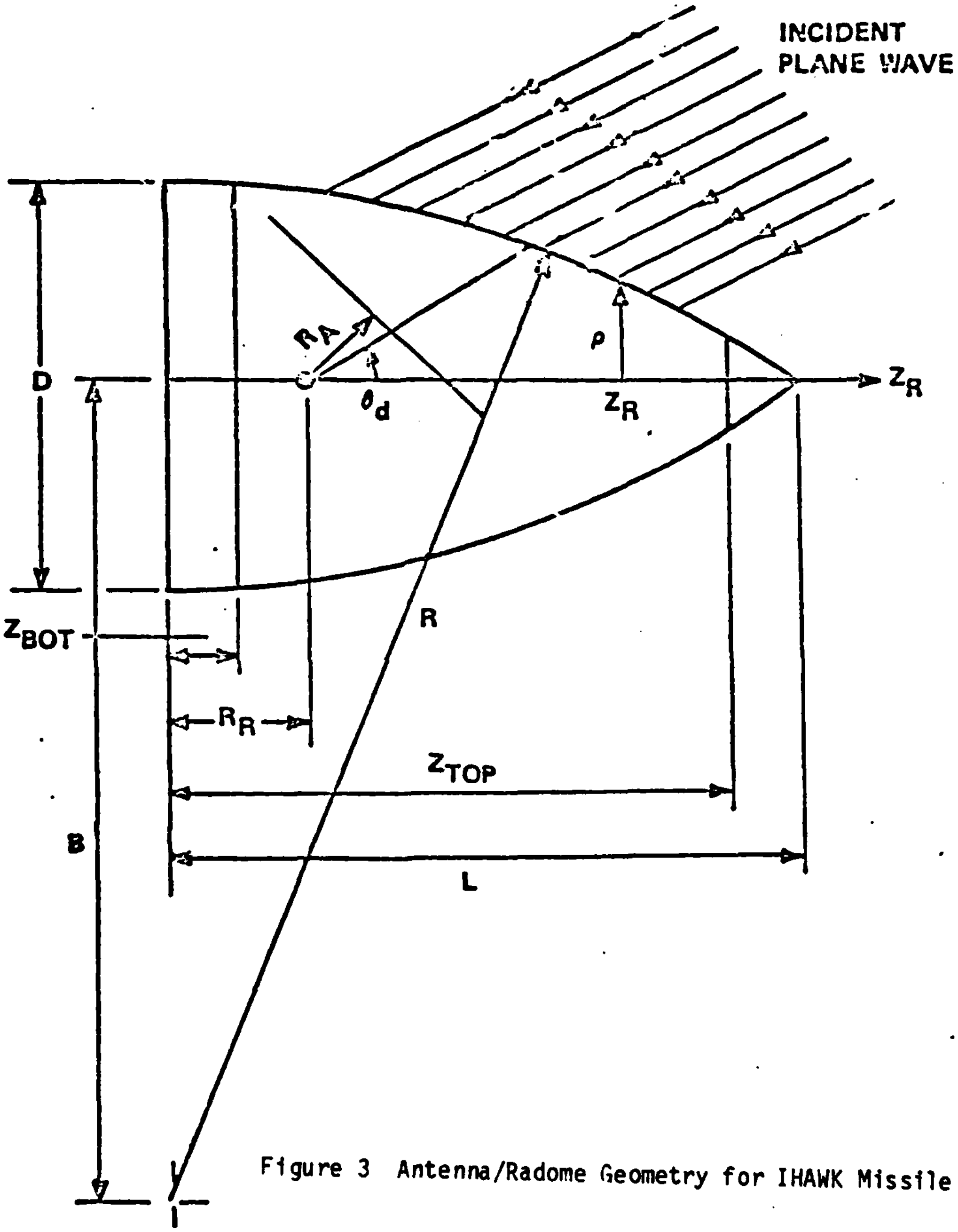


Figure 3 Antenna/Radome Geometry for IHAWK Missile

6.0 CONCLUSIONS AND RECOMMENDATIONS

Plots resulting from the use of the new offset horn method are accurate, readable, and present a better overall representation of the true boresight error induced by the radome. Software improvements have proven to be reliable and time-saving tools aiding in the performance of radome testing. Due to these software improvements, the possibility now arises to study boresight error slopes. Boresight error slope is an important characteristic since it indicates the rate of change of the boresight error. Knowledge of this rate is significant for missile response since it can be used to predict future boresignt errors by utilizing the information of target movement.

The majority of the recommendations presented in [1] have been implemented. A time-saving recommendation repeated here is the need for a means to transfer data directly from the test system's floppy disks to the data processing system. This modification would also eliminate the need to acquire data cassette tapes. Other recommendations include improved plotting capabilities (possibly on site) and the acquisition of a graphics terminal dedicated to radome testing.

REFERENCES

[1] Tilley, Patrick A. and Gean, Roger G., "Radome Boresight Assessment," UAH Technical Report #82/10, Contract #DAAH01-82-D-A008, Delivery Order #0006, prepared for the Systems Simulation and Development Directorate, U. S. Army Missile Command, Redstone Arsenal, AL, October 1982.

[2] Bassett, H. L., Hallum, M. M., Handley, J. C., Harris, J. N., Huddleston, G. K., Letson, K. N. and Yost, D. J., "Radome and Irdome Technology, A Short Course of Instruction," sponsored by U.S. Army Missile Command, Redstone Arsenal, AL, November 1982.

Appendix A
Boresight Error Plots

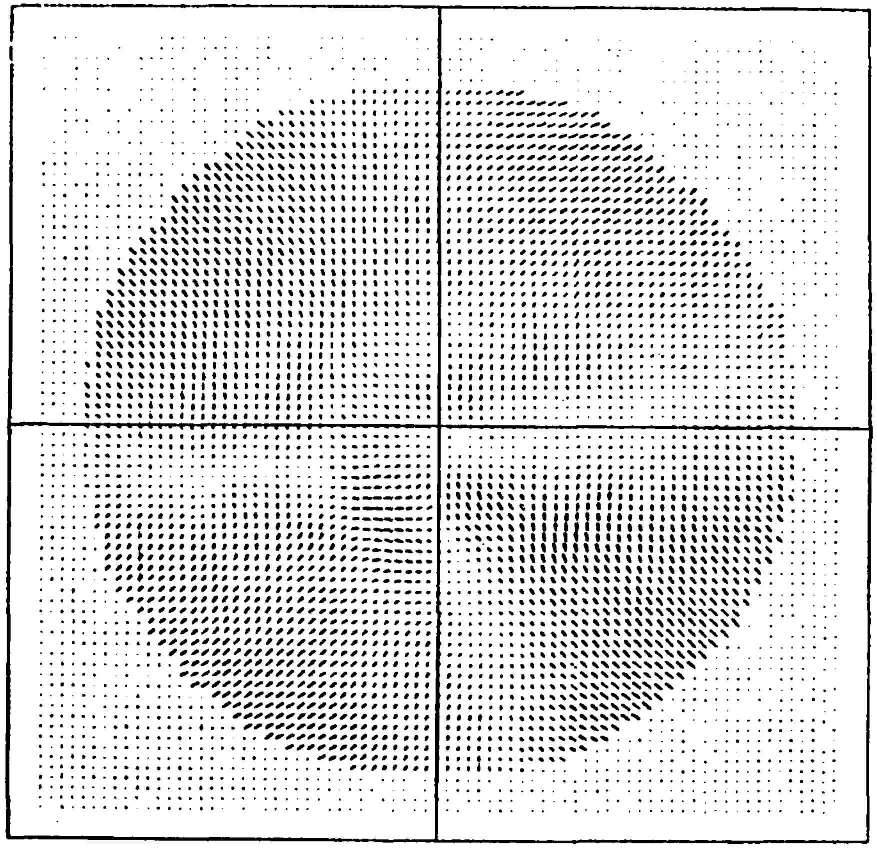


Figure A-1
Vector Plot of Boresight Error for Acceptable IHAWK Radome - Case I
(Scale: Distance between dots = $\frac{1}{2}°$ of error = $1°$ of position)

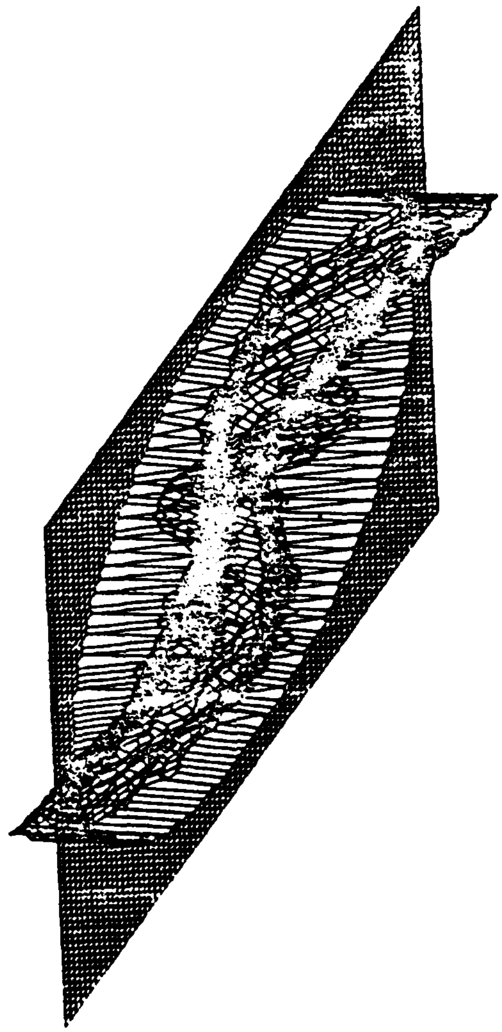


Figure A-2
Carpet Plot of Azimuth Error for Case I

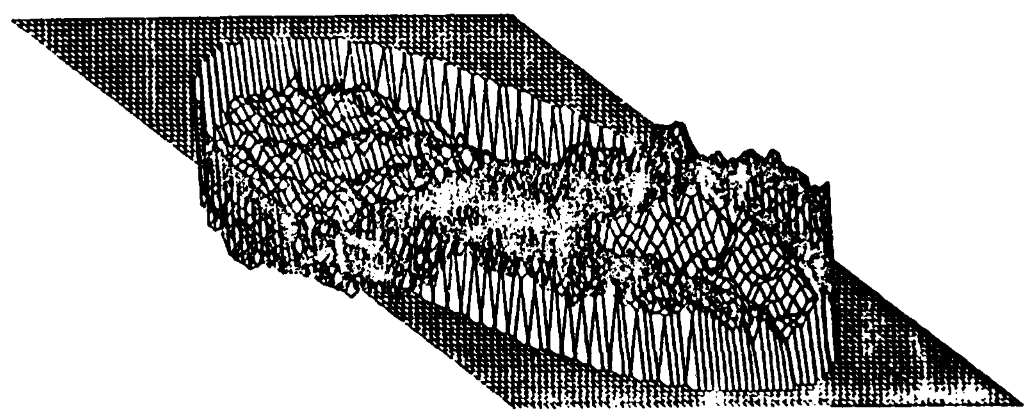


Figure A-3
Carpet Plot of Elevation Error for Case I

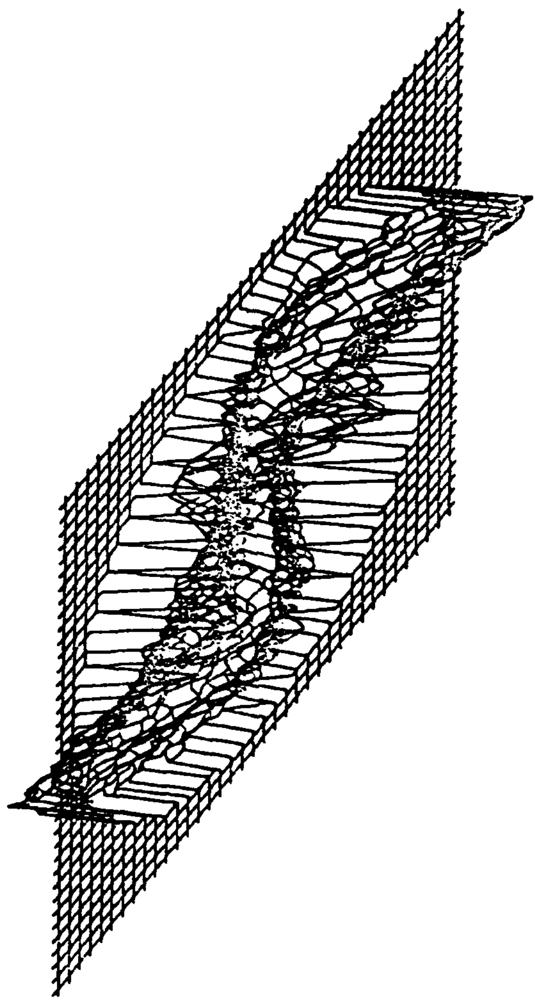


Figure A-4
Carpet II Plot of Azimuth Error for Case I

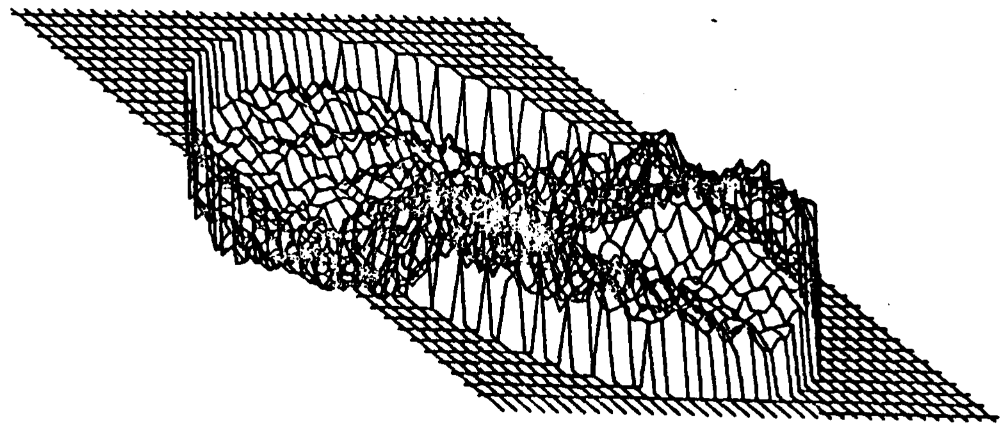


Figure A-5
Carpet II Plot of Elevation Error for Case I

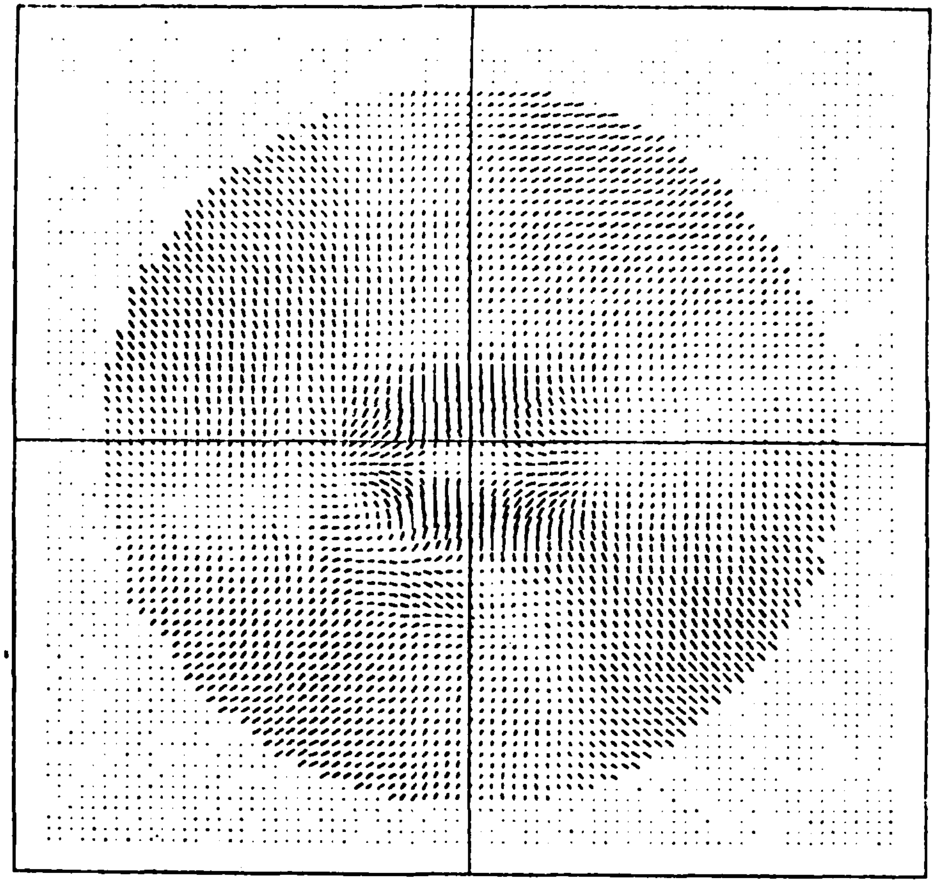


Figure A-6
Vector Plot of Boresight Error for Radome with Plexiglass Obstruction - Case II
(Scale: Distance between dots = $1/2°$ of error = $1°$ of position)

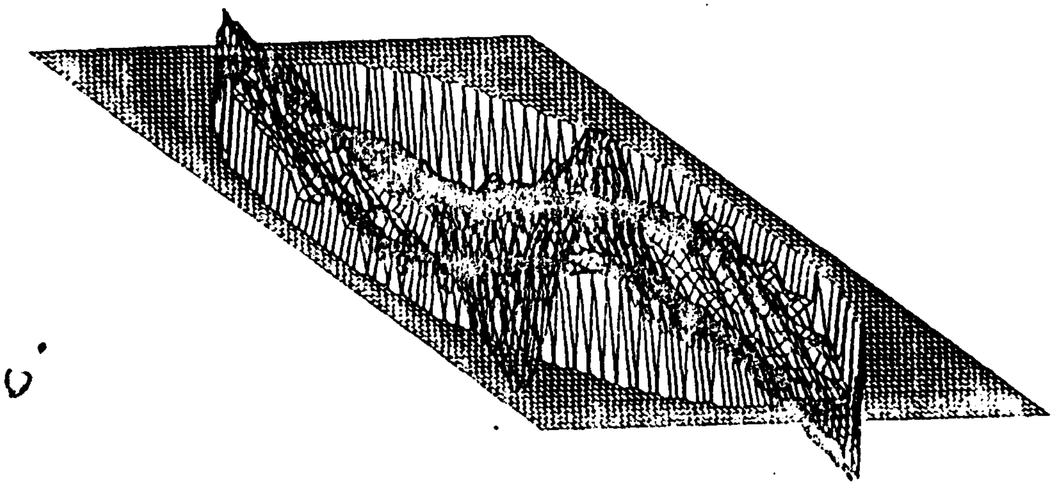


Figure A-7
Carpet Plot of Azimuth Error for Case II

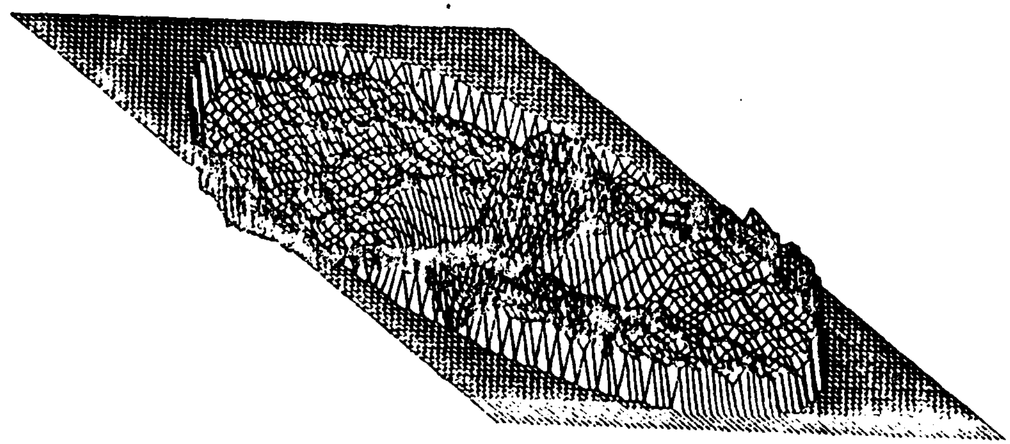


Figure A-8
Carpet Plot of Elevation Error for Case II

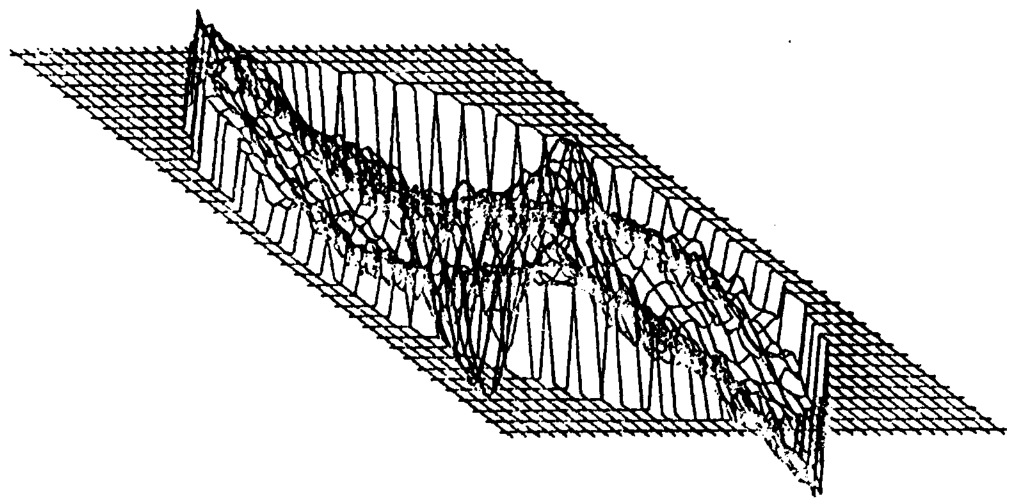


Figure A-9
Carpet II Plot of Azimuth Error for Case II

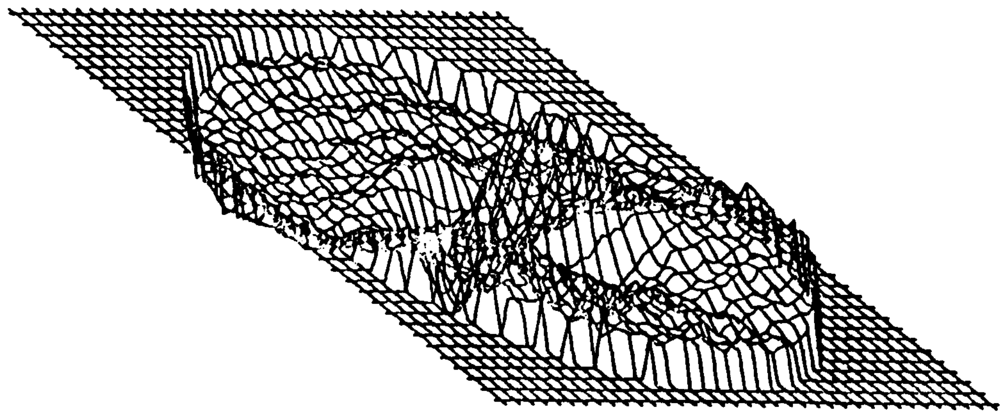


Figure A-10
Carpet II Plot of Elevation Error for Case II

Appendix B
Radome Positioner Software Listing

```
C FORTRAN SOURCE DRIVER FOR RMR-V2.1 TIM PALMER
C INITIALIZATION
        COMMON    ISAMP
        ISAMP     =23.0
1       PRINT 2
        PRINT 4
        PRINT 6
        PRINT 8
2       FORMAT (' THE UNIVERSITY OF ALABAMA IN HUNTSVILLE')
4       FORMAT (' RADOME MEASUREMENTS RECIEVER VERSION 2.1'/)
6       FORMAT ('                 MODIFIED: 10/82')
8       FORMAT ('                 BY: T PALMER'/)
        PRINT 10
        PRINT 12
        PRINT 14
10      FORMAT (' 1) ANTENNA SHOULD BE BORESIGHTED')
12      FORMAT (' 2) POSITIONER GIMBAL SPEEDS SHOULD BE SET')
14      FORMAT (' 3) POSITIONER SHOULD BE AT FIRST SAMPLE POINT'/)
15      PRINT 16
16      FORMAT(' 1=BORESIGHT ANTENNA,2=CONTINUE:')
        READ 18,IDUM
18      FORMAT()
        GO TO (50,20),IDUM
        GO TO 15
20      PRINT 22
22      FOPMAT(/' ENTER SAMPLE SIZE (0.5-2.0):')
        READ 24,SAM
24      FORMAT ()
        IF (SAM .LT. 0.5 .OR. SAM .GT. 2.0) GO TO 20
        ISAMP=SAM*23.0
        PRINT 245
245     FORMAT(/)
26      FORMAT(/' 1=GO,2=ABORT,E (ANY TIME)=ESCAPE:')
        CALL OPENIT
25      PRINT 26
        READ 28,IDUM
28      FORMAT()
        GO TO (30,32),IDUM
        GO TO 25
30      PRINT 305
305     FORMAT(/' TEST NOW ACTIVE..'/)
        CALL CNTL
        PRINT 31
31      FORMAT(' END OF TEST:'/)
        GO TO 1
32      PRINT 34
34      FORMAT(' TEST ABORTED:' )
        GO TO 1
50      PRINT 52
52      FORMAT(' REBOOT SYSTEM,TYPE (LOAD SIG)'///)
        END
```

```
         NAM     CNTL
*
         OPT     REL
*
         XREF    DSCT:TXBUF2,AZBIN,ELBIN,AZP,ELP,FIRSTP,TDAZ,TDEL
         XREF    PSCT:NETANL,RADLO,AIM,INIT,WRITER,CLOSIT
         XDEF    CNTL
*
         CSCT
*
ISAMP    RMB     2
         DSCT
ABSAZ    RMB     2
ABSEL    RMB     2
AZCT     RMB     2
ELCT     RMB     2
LSAZ     RMB     2
LSEL     RMB     2
*
*
         PSCT
*
*
CNTL     JSR     INIT
         JSR     AIM              *GET ANGLES INTO AZBIN,ELBIN
TSTLP    LDA A   0FCF5H           *CHECK KEYBOARD FOR AN "E"
         CMP A   #45H
         BNE     SMPL
         JMP     TSTND
SMPL     JSR     AIM
         LDX     AZBIN
         STX     ABSAZ
         LDX     ELBIN
         STX     ABSEL
TAZ      LDA A   AZBIN
         AND A   #08              *CHECK FOR NEGATIVE ANGLE
         BEQ     PAZ              *GO IF POSITIVE
NAZ      LDA A   #00
         LDA B   #00              *GET ABS VALUE OF AZ
         SUB A   AZBIN+1
         SBC B   AZBIN
         STA A   ABSAZ+1
         STA B   ABSAZ
PAZ      LDA A   ABSAZ+1
         LDA B   ABSAZ
         LDX     #0000
         STX     TDAZ
         STX     AZCT
         JSR     AZDLP
         LDA A   ABSAZ+1
         LDA B   ABSAZ
         SUB A   TDAZ+1
         SBC B   TDAZ
         TST B
         BNE     EXT
         CMP A   #01
         BGT     EXT
         LDA A   AZBIN+1
         LDA B   AZBIN
         SUB A   LSAZ+1
         SBC B   LSAZ
         BMI     LSGTR
         BPA     AZGTR
LSGTR    LDA A   LSAZ+1
```

```
         LDA B    LSAZ
         SUB A    AZBIN+1
         SBC B    AZBIN
         CMP A    #03
         BGT      GSMP
         BRA      EXT
AZGTR    LDA A    AZBIN+1
         LDA B    AZBIN
         SUB A    LSAZ+1
         SBC B    LSAZ
         CMP A    #03
         BGT      GSMP
         BRA      EXT
EXT      JMP      TSTLP
GSMP     LDX      AZBIN
         STX      AZP
         STX      LSAZ
         LDX      ELBIN
         STX      ELP
         STX      LSEL
         JSR      NETANL
         JSR      RADLO
         LDX      #TXBUF2+2
         LDA B    #22
         JSR      WRITER
         JMP      TSTLP
TSTND    JSR      CLOSIT
         RTS
AZDLP    SUB A    ISAMP+1
         SBC B    ISAMP
         BMI      DSAZ
         PSH A
         PSH B
         LDA A    AZCT+1
         LDA B    AZCT
         ADD A    #01
         ADC B    #00
         STA A    AZCT+1
         STA B    AZCT
         PUL B
         PUL A
         BRA      AZDLP
DSAZ     LDA A    AZCT+1
         LDA B    AZCT
         SUB A    #01
         SBC B    #00
         STA A    AZCT+1
         STA B     AZCT
         BMI      DON2
         PSH A
         PSH B
         LDA A    TDAZ+1
         LDA B    TDAZ
         ADD A    ISAMP+1
         ADC B    ISAMP
         STA A    TDAZ+1
         STA B    TDAZ
         PUL B
         PUL A
         BRA      DSAZ
DON2     RTS
         END
```

```
*
*
         NAM    INIT
*
*
         OPT    REL
*
*
         XREF   ANY:CRB,DDRB,SRCR1,SRCR2
         XDEF   INIT
*
*
         PSCT
*
*
INIT     CLR    CRB     *INITIALIZE PIA
         LDA A  #0FFH
         STA A  DDRB
         LDA B  #04H
         STA B  CRB
         STA A  DDRB
         LDA A  #03H    *INITIALIZE ACIA'S
         STA A  SRCR1
         STA A  SRCR2
         LDA A  #31H
         STA A  SRCR1
         LDA A  #01H
         STA A  SRCR2
         RTS
*
*
         END
*
*
```

```
*
        NAM     AIM
*
        OPT     REL
*
        XDEF    AIM
*
        XREF    ANY:DATA,MUX,GAIN,CONVRT,STATUS
*
        XREF    DSCT:GAIN1,CHAN,AZBIN,ELBIN
*
**      SUBROUTINE TO READ THE AZ AND EL ANGLES FROM ANALOG PORTS
**      ANGLES ARE ON CHAN 3 FOR AZ AND 4 FOR EL
*
        PSCT
AIM     LDA A   GAIN1    *SET A/D GAIN
        STA A   GAIN
        CLR     CHAN    *CLR CHAN POSITION
        LDA A   #02     *LOAD MUX TO 02 TO READ CHAN 3
        STA A   MUX
        LDA A   CHAN    *INITIALIZE CHAN TO 3
        ADD A   #3
BBACK   STA A   CHAN
        STA A   CONVRT  *START CONVERSION PROCESS
CKSTAT  LDA A   STATUS  *WAIT UNTIL STATUS READY
        BPL     CKSTAT
        LDA A   CHAN
        CMP A   #4      *CHAN 3 HAS BEEN READ READ CHAN 4
        BEQ     CHAN4
        LDX     DATA    *READ AND STORE AZ ANG
        STX     AZBIN
        LDA A   CHAN    *INCREMENT CHAN TO 4
        ADD A   #1
        STA A   CHAN
        INC     MUX     *INCREMENT MUX TO 03 TO READ CHAN 4
        BRA     BBACK   *BRANCH BACK TO READ CHAN 4 THE EL ANG
CHAN4   LDX     DATA
        STX     ELBIN
        RTS
        END
```

```
        NAM     NETANL

        OPT     REL

        XREF    ANY:GAIN,STATUS,DATA,CONVRT,MUX,DDRB
        XREF    DSCT:SAVEX,GAIN4,SWITCH,WHICH,FOUR,OFFSET,SWILOC,SWIPOS
        XREF    DSCT:CHAN,TIME,TWO,OUTLOC
        XREF    PSCT:COMPUT,WAIT
        XDEF    NETANL

        PSCT

*       "NETANL" SUBROUTINE
*       CONTROLS SAMPLING OF SUM, AZIMUTH, AND ELEVATION DATA.  ALSO CONTROLS
*       THE CONVERSION OF THIS DATA INTO DIGITAL FORM.
*       EXIT:   DIGITAL (TWO'S COMPLIMENT) DATA IS STORED IN THE ORDER IT
*               WAS TAKEN UNDER THE FOLLOWING LABELS;
*               AMPSUM, PHASUM, AMPAZ, PHAAZ, AMPEL, PHAEL

ETANL   PSH A           * SAVE ACC A
        PSH B           * SAVE ACC B
        STX     SAVEX   * SAVE X REGISTER

        LDA A   GAIN4   *SET GAIN OF ADC AMP
        STA A   GAIN


        CLR     SWITCH  * SET SWITCH MEMORY OUTPUT POINTER TO ZERO
        CLR     WHICH   * SET SWITCH POSITION POINTER TO ZERO

EXT     INC     WHICH   * INCREMENT SWITCH POSITION POINTER

        LDA A   WHICH   * LOAD ACC A WITH SWITCH POSITION
        CMP A   FOUR    * CHECK IF SWITCH POSITION IS STILL VALID
        BEQ     RESET   * IF NOT, RESET SWITCH

        STA A   OFFSET  * STORE OFFSET = WHICH TO COMPUTE NEW ADDRESS

        LDX     SWILOC  * LOAD X REGISTER WITH ADDRESS TO BE CHANGED

        JSR     COMPUT  * JUMP TO ROUTINE TO COMPUTE NEW ADDRESS

        LDA A   0,X     * LOAD THE CODE FROM THE COMPUTED ADDRESS IN ACC A

        STA A   DDRB    * SEND THE CODE TO THE PIA TO SWITCH THE SWITCH

        CLR     CHAN    * SET CHANNEL POINTER TO ZERO
        CLR     MUX     * SET ACTUAL CHANNEL TO ZERO

        LDX     TIME    * LOAD X REGISTER WITH TIME CONSTANT FOR WAIT ROUTINE
        JSR     WAIT    * GO TO WAIT SUBROUTINE

        BPA     CHANN1  * ONLY DO FOLLOWING SECTION WHEN GETTING DATA [illegible]
```

```
*
CHANN3   INC      MUX       * INCREMENT ACTUAL CHANNEL
         LDA A    CHAN      * LOAD ACC A WITH CURRENT CHANNEL POINTER
         ADD A    TWO       * INCREMENT BY TWO
         STA A    CHAN      * STORE NEW POINTER
*
CHANN1   STA A    CONVRT    * START CONVERSION PROCESS BY WRITING INTO MEMORY
*
CKSTAT   LDA A    STATUS    * CHECK STATUS UNTIL READY
         BPL      CKSTAT    * WHEN READY CONTINUE
*
*
*
         LDA A    SWITCH    * LOAD ACC A WITH SWITCH MEMORY POINTER
         ADD A    CHAN      * ADD CHANNEL POINTER
         STA A    OFFSET    * THIS IS THE OFFSET USED TO COMPUTE THE OUTPUT ADDRESS
*
         LDX      OUTLOC    * LOAD THE X REGISTER WITH THE ADDRESS TO BE CHANGED
*
         JSR      COMPUT    * JUMP TO THE ROUTINE TO COMPUTE THE NEW ADDRESS
*
*
         LDA A    DATA      * GET FIRST BYTE OF DATA
         STA A    0,X       * STORE IN PREDETERMINED POSITION
*
         INX                * INCREMENT OUTPUT ADDRESS
*
         LDA A    DATA+1    * GET SECOND BYTE OF DATA
         STA A    0,X       * STORE
*
         LDA A    CHAN      * CHECK IF ONLY CHANNEL 1 HAS BEEN DONE
         BEQ      CHANN2    * IF SO, GO DO CHANNEL 2
*
         LDA A    SWITCH    * LOAD ACC A WITH CURRENT SWITCH MEMORY POINTER
         ADD A    FOUR      * INCREMENT BY FOUR
         STA A    SWITCH    * STORE NEW POINTER
*
*
         BRA      NEXT      * REDO ROUTINE FOR NEXT SWITCH POSITION
*
*
RESET    LDA A    SWIPOS    * LOAD ACC A WITH CODE TO TURN SWITCH OFF
         STA A    DDRB      * TURN SWITCH OFF
*
*
         LDX      SAVEX     * RESTORE X REGISTER
         PUL B              * RESTORE ACC B
         PUL A              * RESTORE ACC A
*
*
         RTS                * RETURN TO CALLING ROUTINE
*
         END
```

```
*       NAM     COMPUT
*       OPT     REL
*
*
        XDEF    COMPUT
        XREF    DSCT:LOC,OFFSET,ZERO
*
*
        PSCT
*
*
**      "COMPUT" SUBROUTINE
**      ADDS AN EIGHT BIT NUMBER (OFFSET) TO A SIXTEEN BIT NUMBER (ADDRESS)
**      ENTRY:    X REGISTER -- ADDRESS
**         /      OFFSET -- OFFSET
**      EXIT:     X REGISTER -- MODIFIED ADDRESS
**      A AND B ACCUMULATORS ARE SAVED
*
*
COMPUT  STX     LOC       * TEMPORARILY STORE ADDRESS TO BE CHANGED
*
        PSH A             * SAVE ACC A
        PSH B             * SAVE ACC B
*
        LDA A  LOC        * LOAD ACC A WITH MS BYTE OF ADDRESS TO BE CHANGED
        LDA B  LOC+1      * LOAD ACC B WITH LS BYTE OF ADDRESS TO BE CHANGED
*
        ADD B  OFFSET     * ADD OFFSET TO LS BYTE
        ADC A  ZERO       * CARRY IF NECESSARY
*
        STA A  LOC        * TEMPORARILY STORE NEW ADDRESS
        STA B  LOC+1      *
*
        LDX     LOC       * LOAD THE X REGISTER WITH THE.NEW ADDRESS
*
        PUL B             * RESTORE ACC B
        PUL A             * RESTORE ACC A
*
        RTS               * RETURN TO CALLING PROGRAM
*
        END
```

```
*
        NAM     WAIT
*
        OPT     REL
*
*
        XDEF    WAIT
        PSCT
*
*
**      "WAIT" SUBROUTINE
**      STALLS (TIME) X (100 MICROSECONDS) WHERE TIME = CONTENTS OF X REGISTER
**      ACCUMULATORS A AND B ARE NOT AFFECTED
*
*
WAIT    PSH A             * SAVE CONTENTS OF ACC A
*
        LDA A   #0BH      * LOAD ACC B WITH INITIAL COUNTDOWN
*
WAIT1   DEC A             * DECREMENT FOR COUNTDOWN
        BNE     WAIT1     * REDO IF COUNTDOWN IS NOT FINISHED
*
        LDA A   #0FH      * LOAD ACC A WITH SUBSEQUENT COUNTDOWN
*
        DEX               * DECREMENT TIME
        BNE     WAIT1     * USE MORE TIME IF TIME > 0
*
        CMP A   0,X       * STALL FOR 5 MICROSECONDS
*
        PUL A             * RESTORE CONTENTS OF ACC A
*
        RTS               * RETURN TO CALLING ROUTINE
*
        END
```

```
        NAM     RADLO

        OPT     REL

        XREF    DSCT:HEADER,FINAL
        XREF    DSCT:G1,BSEAZ,BSEEL,AZBIN,ELBIN
        XREF    DSCT:AMPSUM,PHASUM,AMPAZ,PHAAZ,AMPEL,PHAEL
        XREF    DSCT:TXBUF2
        XREF    ANY:G1MSBP,G1LSBP,BAMSBP,BALSBP,BEMSBP,BELSBP
        XREF    ANY:ABMSBP,ABLSBP,EBMSBP,EBLSBP
        XREF    ANY:ASMSBP,ASLSBP,PSMSBP,PSLSBP
        XREF    ANY:AAMSBP,AALSBP,PAMSBP,PALSBP
        XREF    ANY:AEMSBP,AELSBP,PEMSBP,PELSBP
        XREF    ANY:DACA.,DACA2,DACA3,DACA4,DACA5
        XDEF    RADLO

        PSCT

*       "RADLO" (1) LOADS THE RADOME MEASUREMENTS
*                   INTO THE BUFFER FOR THE RFSS INTERFACE
*               (2) LOADS THE DAC'S

ADLO    LDX     HEADER          *LOAD HEADER AT TOP OF BUFFER
        STX     TXBUF2
        LDX     FINAL           *LOAD FINAL AT BOTTOM OF BUFFER
        STX     TXBUF2+24
        LDX     #TXBUF2         *LOAD BUFFER STARTING ADDRESS INTO X

        LDA A   G1              *LOAD BUFFER ACCORDING TO
                                 PREDETERMINED BYTE POSITIONS
        LDA B   G1+1
        STA A   2,X
        STA B   3,X
        LDA A   BSEAZ
        LDA B   BSEAZ+1
        STA A   4,X
        STA B   5,X
        LDA A   BSEEL
        LDA B   BSEEL+1
        STA A   6,X
        STA B   7,X
        LDA A   AZBIN
        LDA B   AZBIN+1
        STA A   8,X
        STA B   9,X
        LDA A   ELBIN
        LDA B   ELBIN+1
        STA A   10,X
```

```
         STA B   11,X
         LDA A   AMPSUM
         LDA B   AMPSUM+1
         STA A   12,X
         STA B   13,X
         LDA A   PHASUM
         LDA B   PHASUM+1
         STA A   14,X
         STA B   15,X
         LDA A   AMPAZ
         LDA B   AMPAZ+1
         STA A   16,X
         STA B   17,X
         LDA A   PHAAZ
         LDA B   PHAAZ+1
         STA A   18,X
         STA B   19,X
         LDA A   AMPEL
         LDA B   AMPEL+1
         STA A   20,X
         STA B   21,X
         LDA A   PHAEL
         LDA B   PHAEL+1
         STA A   22,X
         STA B   23,X
*
*
*
         LDX     G1
         STX     DACA1
         LDX     BSEAZ
         STX     DACA2
         LDX     AZBIN
         STX     DACA3
         LDX     BSEEL
         STX     DACA4
         LDX     ELBIN
         STX     DACA5
*
*
         RTS
*
         END
```

```
*
         NAM     SDATA
*
         OPT     REL
*
         XDEF    DEGRAD,TEN,TWENTY,CON1,CON2,CTS,TMPX2,FIRSTP
         XDEF    AZP,ELP,ICALPT,DEG,PANEL,MANEL,PANAZ,MANAZ
         XDEF    G1,BSEAZ,BSEEL,REFSUM,SUMDB,AZDB,ELDB,G0,KEL,KAZ
         XDEF    DACA1,DACA2,DACA3,DACA4,DACA5,G0MSBP,G0LSBP
         XDEF    KAMSBP,KALSBP,KEMSBP,KELSBP,SDMSBP,SDLSBP,ADMSBP,ADLSBP
         XDEF    EDMSBP,EDLSBP,G1MSBP,G1LSBP,BAMSBP,BALSBP,BEMSBP,BELSBP
         XDEF    ABP,EBP,TEMPX1
         XDEF    ABMSBP,ABLSBP,EBMSBP,EBLSBP
         XDEF    ASMSBP,ASLSBP,PSMSBP,PSLSBP
         XDEF    AAMSBP,AALSBP,PAMSBP,PALSBP
         XDEF    AEMSBP,AELSBP,PEMSBP,PELSBP
         XDEF    SFIA,SFIP,SFG1,SFBE
         XDEF    TDAZ,TDEL,HEADER,FINAL
        XDEF    TMPSIN
*
         DSCT
*
DEGRAD   FDB     788EH      *DEGREES TO RADIANS CONVERSION FACTOR (.0174)
         FDB     0FA35H
TEN      FDB     10         *THE NUMBER TEN (10)
TWENTY   FDB     20         *THE NUMBER TWENTY (20)
CON1     FDB     00180H      *  (.05)
         FDB     00000
CON2     FDB     058BH      *DEGREES TO MILLI-RADIANS CONVERSION FACTOR (17.4)
         FDB     0A057H
CTS      RMB     1
TEMPX1   RMB     2
TMPX2    RMB     2
DEG      FDB     00073H         *DATA POINT 7, DEGREES OF MAXIMUM EL AND AZ
G0       RMB     4
KEL      FDB     00180H
 FDB 00000
KAZ      FDB     00180H
 FDB 00000
PANEL    FDB     05A0H          *ERROR VOLTAGE AT +DEG ELEVATION
         FDB     0000H
MANEL    FDB     05A0H          *ERROR VOLTAGE AT -DEG ELEVATION
         FDB     0000H
PANAZ    FDB     05A0H          *ERROR VOLTAGE AT +DEG AZIMUTH
         FDB     0000H
MANAZ    FDB     04A0H          *ERROR VOLTAGE AT -DEG AZIMUTH
         FDB     0000H
G1       RMB     2          *TRANSMISSION LOSS, RADOME LOOKING AT ANGLE
BSEAZ    RMB     2          *BORESIGHT ERROR AT ANGLE (AZIMUTH)
BSEEL    RMB     2          *BORESIGHT ERROR AT ANGLE (ELEVATION)
REFSUM   FDB     04A0H
         FDB     0000H
SUMDB    RMB     4
AZDB     RMB     4
ELDB     RMB     4
FIRSTP   RMB     1
AZP      RMB     2
ELP      RMB     2
TDAZ     RMB     2
TDEL     RMB     2
ICALPT   RMB     1
ABP      FCB     5
EBP      FCB     6
HEADER   FDB     0FFFFH
```

```
FINAL     FDB     0FF0DH
GOMSBP    FCB     1
G0LSBP    FCB     2
KAMSBP    FCB     3
KALSBP    FCB     4
KEMSBP    FCB     6
KELSBP    FCB     7
SDMSBP    FCB     1
SDLSBP    FCB     2
ADMSBP    FCB     3
ADLSBP    FCB     4
EDMSBP    FCB     6
EDLSBP    FCB     7
G1MSBP    FCB     2
G1LSBP    FCB     3
BAMSBP    FCB     4
BALSBP    FCB     5
BEMSBP    FCB     6
BELSBP    FCB     7
ABMSBP    FCB     8
ABLSBP    FCB     9
EBMSBP    FCB     10
EBLSBP    FCB     11
ASMSBP    FCB     12
ASLSBP    FCB     13
PSMSBP    FCB     14
PSLSBP    FCB     15
AAMSBP    FCB     16
AALSBP    FCB     17
PAMSBP    FCB     18
PALSBP    FCB     19
AEMSBP    FCB     20
AELSBP    FCB     21
PEMSBP    FCB     22
PELSBP    FCB     23
*
*
SFIA      FDB     006A3H
          FDB     0C28FH
SFIP      FDB     00483H
          FDB     0020CH
SFG1      FDB     00788H
          FDB     04000H
SFBE      FDB     00BFFH
          FDB     0E000H
TMPSIN    RMB     4
*
          ASCT
*
**         DAC LOCATIONS
*
DACA1     EQU     0E02EH
DACA2     EQU     0E02CH
DACA3     EQU     0E028H
DACA4     EQU     0E03EH
DACA5     EQU     0E03AH
*
*
          END
*
*
```

```
*
          NAM    RDATA
*
          OPT    REL
*
*
          XDEF   TEMPA,TEMPB,TEMPX,BCDSGN,BINUPR,OUTBUF
          XDEF   AZBCDS,AZBCD,ELBCDS,ELBCD,AZBIN,ELBIN,TMPBCD
          XDEF   TMPBUF,RXBUF1,TXBUF1,TXBUF2,RXBU1P,TXBU2P
          XDEF   AZANG,ELANG,PLUS,APST,DCML,CR,AINTR,TXDRE
          XDEF   SRCR1,RTDR1,SRCR2,RTDR2
          XDEF   APUDAT,APUSTA,DDRA1
*
*
          DSCT
PLUS      FCB    2BH
APST      FCB    27H
DCML      FCB    2EH
CR        FCB    0DH
TXDRE     FCB    02H
*
*
**      ALLOCATED MEMORY AREA.
*
**      TEMPORARY MEMORY VARIABLES.
*
TEMPA     RMB    1             *ACCUMULATOR A (BCDBIN,BINANG).
TEMPB     RMB    1             *ACCUMULATOR B (BINANG).
TEMPX     RMB    2             *INDEX REGISTER (MOVBUF,BCDANG).
BCDSGN    RMB    1             *ASCII SIGN OF BCD ANGLE (BINANG,BCDANG,ANGLE).
BINUPR    RMB    1             *USED IN FINDING MS BYTE OF BINARY VALUE (BCDBIN).
OUTBUF    RMB    2             *POINTER TO TEMPORARY OUTPUT BUFFER (MOVBUF).
*
          DSCT
*
*
**      DEDICATED MEMORY VARIABLES.
*
AZBCDS    RMB    1             *ASCII SIGN OF ELEVATION BCD ANGLE (ANGLE).
AZBCD     RMB    2             *PACKED BCD FORM OF AZIMUTH ANGLE (ANGLE).
ELBCDS    RMB    1             *ASCII SIGN OF ELEVATION BCD ANGLE (ANGLE).
ELBCD     RMB    2             *PACKED BCD FORM OF ELEVATION ANGLE (ANGLE).
AZBIN     RMB    2             *TWO'S COMPLIMENT OF AZIMUTH ANGLE (ANGLE).
ELBIN     RMB    2             *TWO'S COMPLIMENT OF ELEVATION ANGLE (ANGLE).
TMPBCD    RMB    1             *USED IN PACKING BCD ANGLE (BCDPCK).
RXBU1P    RMB    2             *POINTER FOR ASCII CHAR IN RCVR BUFFER 1 (RCVRFP).
TXBU2P    RMB    2             *POINTER FOR ASCII CHAR IN TXMT BUFFER 2 (TXMTAI).
*
*
*
**    EXTERNAL INTERFACE BUFFER AREA
*
*
```

```
TMPBUF    RMB     20          *TEMPORARY BUFFER AREA (BCDANG)
AZANG     EQU     TMPBUF      *LOCATION IN RXBUF OF AZIMUTH BCD ANGLE (ANGLE)
ELANG     EQU     TMPBUF+10   *LOCATION IN RXBUF OF ELEVATION BCD ANGLE(ANGLE)
RXBUF1    FDB     0000H       *ASCII CHARS REC'D FROM POSITIONER (BCDANG,RCVPRP)
          FDB     0000H
          FDB     0000H
          FDB     002EH
          FDB     0027H
          FDB     0000H
          FDB     0000H
          FDB     0000H
          FDB     002EH
          FDB     0027H
TXBUF1    RMB     20
TXBUF2    RMB     26
*
*
          ASCT
*
**    I/O DEVICE EQUATE SYMBOLS
*
*
SRCR1     EQU     0E004H
RTDR1     EQU     0E005H
SRCR2     EQU     0E008H
RTDR2     EQU     0E009H
*
**    APU DATA
*
APUDAT    EQU     0E00CH
APUSTA    EQU     0E00DH
DDRA1     EQU     0E000H
AINTP     EQU     0FFF8H
*
          END
```

```
*
         NAM     DDATA
*
         OPT     REL
*
*
         XDEF    SAVEX,LOC,SWITCH,WHICH,CHAN,OFFSET
         XDEF    SWILOC,SWIPOS,CRA
         XDEF    OUTLOC,AMPSUM,PHASUM,AMPAZ,PHAAZ,AMPEL,PHAEL
         XDEF    DDRB,CRB,BASE,GAIN,MUX,CONVRT,STATUS,DATA
         XDEF    GAIN1,GAIN2,GAIN4,GAIN8
         XDEF    TIME,ZERO,TWO,FOUR
*
*
         DSCT
GAIN1    FCB     00H
GAIN2    FCB     01H
GAIN4    FCB     02H
GAIN8    FCB     03H
TIME     FDB     0002H
ZERO     FCB     00H
TWO      FCB     02H
FOUR     FCB     04H
*
*
**       ALLOCATED MEMORY AREA
*
**       TEMPORARY MEMORY VARIABLES
*
SAVEX    RMB     2          * USED TO SAVE CONTENTS OF X REGISTER
LOC      RMB     2          * USED WHEN COMPUTING ADDRESS PLUS OFFSET
SWITCH   RMB     1          * USED IN INCREMENTING PAGE ZERO MEMORY POSITION
*                             (SOFTWARE SWITCH)
WHICH    RMB     1          * POINTER USED TO KEEP TRACK OF SWITCH POSITION
CHAN     RMB     1          * POINTER USED TO KEEP TRACK OF CHANNEL SELECTION
OFFSET   RMB     1          * COMPUTED OFFSET USED WHEN COMPUTING NEW ADDRESS
*
**       DEDICATED MEMORY VARIABLES
```

```
*
SWILOC   FDB    SWIPOS    * LOCATION OF SWITCH POSITION CODES
SWIPOS   FCB    0FFH      * SWITCH POSITION -- OFF
         FCB    0FEH      * SWITCH POSITION -- ONE
         FCB    0FDH      * SWITCH POSITION -- TWO
         FCB    0FBH      * SWITCH POSITION -- THREE
         FCB    0F7H      * SWITCH POSITION -- FOUR
*
**       EXTERNAL INTERFACE BUFFER AREA
*
OUTLOC   FDB    AMPSUM    * LOCATION OF MEMORY RESERVED FOR OUTPUT
AMPSUM   FDB    0001H     * SUM AMPLITUDE
PHASUM   FDB    0001H     * SUM PHASE
AMPAZ    FDB    0001H     * AZIMUTH AMPLITUDE
PHAAZ    FDB    0001H     * AZIMUTH PHASE
AMPEL    FDB    0001H     * ELEVATION AMPLITUDE
PHAEL    FDB    0001H     * ELEVATION PHASE
*
**    EXTERNAL INTERFACE BUFFER AREA
*
CRA      RMB    2         *TERMINATES CONTROL BASED ON AN "S"
         ASCT
DDPB     EQU    0E002H    * DATA DIRECTION REGISTER PIA PORT B
*                           (SWITCH CONTROL)
CRB      EQU    0E003H    * CONTROL REGISTER PIA PORT B
BASE     EQU    0E040H    * BEGINNING OF ADC BOARD " MEMORY " LOCATIONS
GAIN     EQU    BASE+9H   * MEMORY LOCATION TO ACCESS GAIN
MUX      EQU    BASE+0AH  * MEMORY LOCATION TO ACCESS MULTIPLEXER CHANNEL SELECT
CONVRT   EQU    BASE+0EH  * MEMORY LOCATION TO ACCESS CONVERT COMMAND
STATUS   EQU    BASE+0CH  * MEMORY LOCATION TO ACCESS STATUS
DATA     EQU    BASE+0DH  * MEMORY LOCATION TO ACCESS ADC DATA MS BYTE
*
         END
```

```
 NO UNDEFINED SYMBOLS

MEMORY MAP

 S SIZE  STR  END  COMN
 B 0000 0040 0040 0000
 C 0002 2000 2001 0002
 D 035E 2002 235F 0002
 P 1B57 2360 3EB6 000F

MODULE NAME BSCT DSCT PSCT
  MAIN      0040 2002 2360
  CNTL      0040 206C 274E
  INIT      0040 2078 2850
  AIM       0040 2078 2874
  NETANL    0040 2078 28B4
  RADLO     0040 2078 2936
  COMPUT    0040 2078 29D2
  WAIT      0040 2078 29F0
  SDATA     0040 2078 2A00
  RDATA     0040 2108 2A00
  DDATA     0040 217A 2A00
            0040 21A2 2A00
  FTNRUN    0040 2256 2C38

COMMON SECTIONS

  NAME  S SIZE  STR
 .ADDR  P 000F 3EA8
 .ADRDC D 0002 235E

DEFINED SYMBOLS


MODULE NAME: MAIN
  MAIN    P 2360

MODULE NAME: CNTL
  CNTL    P 274E

MODULE NAME: INIT
  INIT    P 2850

MODULE NAME: AIM
  AIM     P 2874

MODULE NAME: NETANL
  NETANL P 28B4

MODULE NAME: RADLO
  RADLO  P 2936

MODULE NAME: COMPUT
  COMPUT P 29D2

MODULE NAME: WAIT
  WAIT   P 29F0

MODULE NAME: SDATA
```

```
  AALSBP D 20EC     AAMSBP D 20EB     ABLSBP D 20E4     ABMSBP D 20E3
  ABP    D 20CB     ADLSBP D 20DA     ADMSBP D 20D9     AELSBP D 20F0
  AEMSBP D 20EF     ASLSBP D 20E8     ASMSBP D 20E7     AZDB   D 20B9
  AZP    D 20C2     BALSBP D 20E0     BAMSBP D 20DF     BELSBP D 20E2
  BEMSBP D 20E1     BSEAZ  D 20AD     BSEEL  D 20AF     CON1   D 2080
  CON2   D 2084     CTS    D 2088     DACA1  A E02E     DACA2  A E02C
  DACA3  A E028     DACA4  A E03E     DACA5  A E03A     DEG    D 208D
  DEGRAD D 2078     EBLSBP D 20E6     EBMSBP D 20E5     EBP    D 20CC
  EDLSBP D 20DC     EDMSBP D 20DB     ELDB   D 20BD     ELP    D 20C4
  FINAL  D 20CF     FIRSTP D 20C1     G0     D 208F     G0LSBP D 20D2
  G0MSBP D 20D1     G1     D 20AB     G1LSBP D 20DE     G1MSBP D 20DD
  HEADER D 20CD     ICALPT D 20CA     KALSBP D 20D4     KAMSBP D 20D3
  KAZ    D 2097     KEL    D 2093     KELSBP D 20D6     KEMSBP D 20D5
  MANAZ  D 20A7     MANEL  D 209F     PALSBP D 20EE     PAMSBP D 20ED
  PANAZ  D 20A3     PANEL  D 209B     PELSBP D 20F2     PEMSBP D 20F1
  PSLSBP D 20EA     PSMSBP D 20E9     REFSUM D 20B1     SDLSBP D 20D8
  SDMSBP D 20D7     SFBE   D 20FF     SFG1   D 20FB     SFIA   D 20F3
  SFIP   D 20F7     SUMDB  D 20B5     TDAZ   D 20C6     TDEL   D 20C3
  TEMPX1 D 2089     TEN    D 207C     TMPSIN D 2103     TMPX2  D 208E
  TWENTY D 207E

MODULE NAME: RDATA
  AINTP  A FFF8     APST   D 2109     APUDAT A E00C     APUSTA A E00D
  AZANG  D 2124     AZBCD  D 2116     AZBCDS D 2115     AZBIN  D 211B
  BCDSGN D 2111     BINUPR D 2112     CR     D 210B     DCML   D 210A
  DDRA1  A E000     ELANG  D 212E     ELBCD  D 2119     ELBCDS D 2118
  ELBIN  D 211D     OUTBUF D 2113     PLUS   D 2108     RTDR1  A E005
  RTDR2  A E009     RXBU1P D 2120     RXBUF1 D 2138     SRCR1  A E004
  SRCR2  A E008     TEMPA  D 210D     TEMPB  D 210E     TEMPX  D 210F
  TMPBCD D 211F     TMPBUF D 2124     TXBU2P D 2122     TXBUF1 D 214C
  TXBUF2 D 2160     TXDRE  D 210C

MODULE NAME: DDATA
  AMPAZ  D 2198     AMPEL  D 219C     AMPSUM D 2194     BASE   A E040
  CHAN   D 2189     CONVRT A E04B     CRA    D 21A0     CRB    A E003
  DATA   A E04D     DDPB   A E002     FOUR   D 2182     GAIN   A E049
  GAIN1  D 217A     GAIN2  D 217B     GAIN4  D 217C     GAIN8  D 217D
  LOC    D 2185     MUX    A E04A     OFFSET D 218A     OUTLOC D 2192
  PHAAZ  D 219A     PHAEL  D 219E     PHASUM D 2196     SAVEX  D 2183
  STATUS A E04C     SWILOC D 218B     SWIPOS D 218D     SWITCH D 2187
  TIME   D 217E     TWO    D 2181     WHICH  D 2188     ZERO   D 2180

MODULE NAME:
  CLOSIT P 2C00     OPENIT P 2AA8     WRITER P 2BD6

MODULE NAME: FTNRUN
  BUF$   D 22B7     COMBA  P 2E4F     EBUF$  D 233C     FILE$  D 22B4
  LPUSE$ D 2297     RUN$   P 3C38     X1$    D 2289     XDKIN$ P 3509
  XDKOT$ P 35CC     XRUND$ P 35CF
```

```
RLOAD
BASE
IDON
LOAD=LIB
LIB=FORLB
MO=MAP1.MO
MAPF
MO=#CN
MAPF
OBJA=OBJ1.LO
LOAD=LIB
LIB=FORLB
EXIT
```

```
MERGE FSTP.RO,TPCNT.RO,INIT.RO:1,AIM.RO:1,COMPUT.RO:1,TODISK.RO:1,LIE.RO
MERGE LIB.RO,NETANL.RO:1,RADLO.RO:1,WAIT.RO:1,DATA.RO:1,LIB.RO
```

Appendix C
Radome Measurements Receiver System Software Listing

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; NEW SCAN PATTERN (#3) SCANS A 32 DEGREE CIRCULAR RASTER PATTERN            ;;
;;                                                                            ;;
;;     ENTRY 1: MUST BE 32.0 OR WHATEVER ELEVATION * -1 TO STOP AT            ;
;;     ENTRY 2: ELEVATION STEP SIZE                                           ;;
;;     ENTRY 3: ANY KEY TO START SCAN                                         ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
ST28     LDX        #0000
         STX        MFLAG       ;CLEAR MOTOR AND PROGRAM FLAGS
         STX        AZKEY       ;FIRST AZIMUTH
         LDX        #3200H      ;RADIUS
         STX        ELKEY       ;NOW ELEVATION
         LDX        #2D2BH      ;"-","+"
         STX        AZKEYS
         LDX        #ST280
         STX        STADDR      ;SAVE RETURN ADDRESS
         DEC        PFLAG       ;SET PROG FLAG
         NOP
ST280    NOP
         NOP
         NOP
ST28A    LDA A      ELSIGN      ;GET CURRENT ELEVATION SIGN
         CMP A      #2BH        ;"+"
         BEQ        ST28C       ;GO IF ELEVATION>0
         LDA A      ELBCD       ;BCD CURRENT ELEVATION
         LDA B      ELBCD+1
         LDX        #PROGA      ;LAST ELEVATION MAGNITUDE
         NOP
         CMP A      #1          ;CHECK PROXIMITY OF LAST POINT
         BHI        ST28AA      ;NOT WITHIN 1 DEGREE
         BRA        FINA        ;WITHIN ONE DEGREE, DONE
ST28AA   TST        CARRY       ;MAYBE OVER 1 DEGREE PAST END
         BMI        ST28B       ;NO,CONTINUE
FINA     LDX        #ST28
         STX        STADDR
         JMP        ST28
ST28B    JSR        NEWAZ       ;CALCULATE NEW AZIMUTH
         LDA A      AZSIGN
         STA A      AZKEYS
         LDX        #ST28B1
         STX        STADDR
         NOP
ST28B1   LDA A      ELKEY+1
         ADD A      PROGB+1
         DAA
         STA A      ELKEY+1
         LDA A      ELKEY
         ADC A      PROGB
         DAA
         STA A      ELKEY
ST28B2   LDX        #ST28B3
         STX        STADDR
         NOP
ST28B3   LDA A      AZSIGN      ;CURRENT AZ SIGN
         CMP A      #2BH        ;"+"
         BEQ        ST28B4
         LDA A      #2BH        ;"+"
         STA A      AZKEYS
         BRA        ST28B5
ST28B4   LDA A      #2DH        ;"-"
         STA A      AZKEYS      ;CHANGES SIGN OF AZIMUTH FOR NEXT POSITION
ST28B5   LDX        #ST28A
         STX        STADDR
```

```
        JMP     ST28A
ST28C   LDA A   ELKEY
        LDA B   ELKEY+1
        LDX     #PROGB
        NOP
        STA A   ELKEY
        STA B   ELKEY+1
        TST     CARRY
        BEQ     ST28C1
        LDA A   #2DH
        STA A   ELKEYS
ST28C1  LDX     #ST28C2
        STX     STADDR
        NOP
ST28C2  JSR     NEWAZ
        LDA A   AZSIGN
        STA A   AZKEYS
        LDX     #ST28C3
        STX     STADDR
        NOP
ST28C3  LDA A   AZSIGN      ;CURR AZ SIGN
        CMP A   #2BH        ;"+"
        BEQ     ST28C4
        LDA A   #2BH
        STA A   AZKEYS
        BRA     ST28C5
ST28C4  LDA A   #2DH
        STA A   AZKEYS
ST28C5  LDX     #ST28A
        STX     STADDR
        JMP     ST28A
;
; "NEWAZ" SUBROUTINE THAT CALCULATES AZ POSITION BASED ON EL.AND PROGA
NEWAZ   LDA A   ELKEY       ;GET ELEVATION MSB
        JSR     BCDBIN      ;GET BINARY EQUIV
        LDA A   TEMPA       ;BINARY EQUIV
        ADD A   TEMPA       ;BINARY EQUIV
        STA A   SAVEA       ;SAVE THIS ONE
        LDX     #TAB1       ;AZ TABLE POINTER
        STX     SAVEX1      ;SAVE POINTER
        LDA A   SAVEX1+1
        LDA B   SAVEX1
        ADD A   SAVEA       ;GET NEW ADDRESS
        ADC B   #00H
        STA A   SAVEX1+1        ;SAVE IT
        STA B   SAVEX1
        LDX     SAVEX1
        LDA A   0,X         ;GET AZIMUTH
        LDA B   1,X         ;GET AZIMUTH LSB
        STA A   AZKEY       ;STORE INTO AZKEY
        STA B   AZKEY+1     ;SAME FOR LSB
        RTS                 ;DONE
TAB1    WORD    03200H
        WORD    03198H
        WORD    03194H
        WORD    03186H
        WORD    03175H
        WORD    03161H
        WORD    03143H
        WORD    03122H
        WORD    03100H
        WORD    03070H
        WORD    03040H
        WORD    03005H
        WORD    02966H
        WORD    02924H
```

```
        WORD    02877H
        WORD    02826H
        WORD    02771H
        WORD    02711H
        WORD    02646H
        WORD    02575H
        WORD    02500H
        WORD    02414H
        WORD    02324H
        WORD    02225H
        WORD    02117H
        WORD    02000H
        WORD    01865H
        WORD    01717H
        WORD    01549H
        WORD    01353H
        WORD    01113H
        WORD    00794H
        WORD    00000H
BCDBIN  CLR     TEMPA
        PSH A           ;SAVE A
        AND A   #0F0H    ;GET UPPER 4 BITS
        LSR A           ;MOVE 1 BIT RIGHT
        LSR A   ;THAT MAKES 2
        LSR A   ;THERES 3
        LSR A   ;THAT"S ALL FOLKS
        CLC             ;CLEAR ANY CARRY
TNLP    TST A
        BEQ     ONELP
        CLR     TEMPA
        PSH A           ;SAVE SHIFTED A (STACK: SHA,A)
LP1     LDA A   TEMPA
        ADD A   #0AH    ;ADD 10
        STA A   TEMPA
        PUL A           ;GET SHIFTED A BACK
        DEC A           ;DEC COUNTER
        BEQ     ONELP           ;IF FINISHED
        PSH A           ;SAVE SHIFTED A AGAIN
        BRA     LP1     ;DO IT AGAIN
ONELP   PUL A           ;PULLS ORIGINAL A
        AND A   #0FH    ;GET LS NIBBLE
        TST A           ;SEE IF DONE
        BEQ     EBBC    ;GO IF NO ONES
        PSH A
LP2     LDA A   TEMPA   ;GET INTERMEDIATE RESULT
        ADD A   #01H    ;INCREMENT BY ONE
        STA A   TEMPA   ;SAVE RESULT
        PUL A           ;GET ONE COUNTER
        DEC A           ;DECREMENT
        BEQ     EBBC    ;GO IF DONE
        PSH A           ;NOT DONE YET
        BRA     LP2
EBBC    RTS
        END
```

```
          SECTION   BLK,ABSOLUTE
          ORG       0000H
DISBUF    BLOCK     20
SIBUF     BLOCK     21
TEMPX     BLOCK     2
RADIUS    BLOCK     2
FPT32     BLOCK     4
ELRESULT  BLOCK     5
AZRESULT  BLOCK     5
INTREG    BLOCK     2
ENTRY1    BLOCK     1
ENTRY2    BLOCK     1
SAVEB     BLOCK     1
SAVEC     BLOCK     1
CHARBF    BLOCK     1
CHARPT    BLOCK     2
CHARCT    BLOCK     2
CHRNUM    BLOCK     1
SAVEA     BLOCK     1
SAVEX     BLOCK     2
SAVEX1    BLOCK     2
TEMPA     BLOCK     1
TEMPB     BLOCK     1
MSBENC    BLOCK     1
LSBENC    BLOCK     1
LETA      BLOCK     1
LETB      BLOCK     1
BCDA      BLOCK     1
BCDB      BLOCK     1
SAVDEC    BLOCK     1
ANGLE     BLOCK     3
SIGN      BLOCK     1
AZSIGN    BLOCK     1
ELSIGN    BLOCK     1
AZBCD     BLOCK     2
ELBCD     BLOCK     2
TEMPX1    BLOCK     2
TEMPX2    BLOCK     2
ENTRYA    BLOCK     1
ENTRYB    BLOCK     1
KEYENT    BLOCK     1
TEMPA1    BLOCK     1
TEMPB1    BLOCK     1
KEYC      BLOCK     1
AZKEY     BLOCK     2
ELKEY     BLOCK     2
AZKEYS    BLOCK     1
ELKEYS    BLOCK     1
MFLAG     BLOCK     1
PFLAG     BLOCK     1
MINUEN    BLOCK     1
SPEEDA    BLOCK     1
SPEEDE    BLOCK     1
AZMAG     BLOCK     2
ELMAG     BLOCK     2
AZEL      BLOCK     1
TEMPS     BLOCK     1
PROGN     BLOCK     1
PROGL     BLOCK     1
PROGA     BLOCK     2
PROGB     BLOCK     2
PROGC     BLOCK     2
KFLAG     BLOCK     1
DFLAGA    BLOCK     1
DFLAGE    BLOCK     1
SFLAGA    BLOCK     1
```

```
SFLAGE    BLOCK     1
TEMPD     BLOCK     2
BCDVSR    BLOCK     2
FPTEL     BLOCK     2
FPTAZ     BLOCK     2
FPTELS    BLOCK     1
FPTAZS    BLOCK     1
PROCNT    BLOCK     1
STADDR BLOCK    2
PROANG    BLOCK     2
BINANG    BLOCK     2
SINE      BLOCK     2
COSINE    BLOCK     2
SSIGN     BLOCK     1
CSIGN     BLOCK     1
SAVE1     BLOCK     1
BINUPR    BLOCK     1
FELLIM    BLOCK     2
NELLIM    BLOCK     2
PAZLIM    BLOCK     2
NAZLIM    BLOCK     2
LFLAGE    BLOCK     1
LFLAGA    BLOCK     1
MSGFLG    BLOCK     1
SAVEX2    BLOCK     2
;                             MODIFICATION V1.3
SPEED     BLOCK     1
GIMSPEED  BLOCK     1
AZSPEED   BLOCK     1
ELSPEED   BLOCK     1
AZSPD     BLOCK     1
ELSPD     BLOCK     1
                    ;
CARRY     BLOCK     1
                    ;
                    ;
                    ;I/O EQUATES
HAFSPD    EQU       0E0H
QUASPD    EQU       0E7H
DDRA2     EQU       08404H    ;MS 4 BITS OF DAC #1-AZIMUTH
CRA2      EQU       08405H
DDRB2     EQU       08406H    ;LS 8 BITS OF DAC #1-AZIMUTH
CRB2      EQU       08407H
DDRA3     EQU       08800H    ;LS 4 BITS OF DAC #2-ELEVATION
CRA3      EQU       08801H
DDRB3     EQU       08802H    ;MS 8 BITS OF DAC #2-ELEVATION
CRB3      EQU       08803H
DISAZ     EQU       0000H
DISEL     EQU       000AH
MSBSEL    EQU       08E03H
LSBSEL    EQU       08E02H
MSBSAZ    EQU       08E01H
LSBSAZ    EQU       08E00H
DDRA      EQU       08400H
CRA       EQU       08401H
DDRB      EQU       08402H
CRB       EQU       08403H
ACIAS     EQU       08408H    ;ACIA STAUS/CONTROL REGISTER
ACIAD     EQU       08409H    ;ACIA DATA REGISTER
APUDATA   EQU       08804H     ;APU DATA INPUT/OUPUT
APUSTAT   EQU       08805H    ;APU COMMAND INPUT AND STATUS OUTPUT
ETHSPD    EQU       0F0H      ;(MOD 1.3)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;      MAIN PROGRAM- 2000-3016 HEX                                 ;;
;;                                                                  ;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        SECTION    MAIN
        ORG        2000H
;
;INITIALIZE PIAS
;
GOE     LDS        #0FFFH
        NOP
        SEI
        LDA A      #3          ;00000011= MASTER RESET
        STA A      ACIAS       ;RESET ACIA
        LDA A      #81H        ;10000001 = 7 BITS, EVEN PARITY, 2 STOP BITS
        STA A      ACIAS       ;SET ACIA FOR RECEIVER INT,TXMIT INT OFF
        CLR        CRA         ;CLEARS CONTROL REG A
        CLR        CRB         ;CLEARS CONTROL REG B
        CLR        CRA2
        CLR        CRB2
;
; ADDTION TO INIT ROUTINE
; COMPENSATION FOR INVERTED BUS DRIVERS
; (MOD 1.2)
;
        LDA A      #0FFH       ;GET OPPOSITE OF 00
        STA A      CRA3        ;CLEAR CONTROL REG OF A SIDE
        STA A      CRB3        ;CLEAR CONTROL REG OF B SIDE
        CLR        DDRB3       ;SETS UP B SIDE OF DATA PORT A ALL OUTPUTS
        LDX        #8400H
        LDA A      #0F0H
        STA A      0,X
        STA A      DDRA3       ;(MOD 1.2) COMPENSATION FOR INV BUS DRIVERS
        LDA A      #7
        STA A      1,X
        LDA A      #0FH
        STA A      0,X
        STA A      4,X
        LDA A      #0FFH
        STA A      2,X
        STA A      6,X
        LDA A      #004H
        STA A      CRA2
        STA A      CRB2
        LDA A      #0FBH       ;(MOD 1.2) PUTS 04 INTO CONTROL REGISTER
        STA A      CRA3
        STA A      CRB3
        LDA A      #03DH
        STA A      CRB         ;SELECTS OUTPUT REGISTER B
;
;     INITIALIZED MOTORS TO ZERO SPEED,ETC.
;
        LDX        #TEMPX
NEXTC   CLR        0,X         ;CLEARS LOCAION POINTED TO BY THE X REGISTER
        INX
        CPX        #CARRY+1
        BNE        NEXTC
;
;ADDITION TO THE INITIALIZATION ROUTINE
;INITIALIZE ASCII FORM OF GIMBAL SPEEDS TO FULL SPEED.
;MOD (1.3)
;
        LDA A      #39H        ;ASCII "9"
        STA A      AZSPEED     ;AZ GIMBAL AT FULL SPEED
        STA A      ELSPEED     ;ELEVATION GIMBAL AT FULL SPEED
;
        LDX        #3500H
        STX        PELLIM
```

```
        STX       NELLIM
        STX       FAZLIM
        STX       NAZLIM
        LDA A     #0DH       ;INITIALIZE (CR) IN SIBUF
        STA A     SIBUF+20
        LDA A     #0FFH
        STA A     LSBSEL
        LDA A     #0FFH
        STA A     LSBSAZ
        STA A     SFLAGA     ;SETS AZ SPEED FLAG TO NOTE ZERO SPEED
        STA A     SFLAGE     ;SETS EL  "     "    "    "    "    "
        LDA A     #001H
        STA A     MSBSAZ      ;TURNS ON POWER TO AZ MOTOR
        STA A     MSBSEL     ;TURNS ON POWER TO EL MOTOR
;
;       INITIALIZES CONTROL LOOP SUCH THAT THE POSTITIONER
;       WILL NOT MOVE UPON POWER-UP  (MOD 1.1)
;
        JSR       SHAENC     ;READ ANGLES
        LDX       AZBCD
        STX       AZKEY      ;UPDATES AZIMUTH KEYENTRY WITH
        LDX       ELBCD      ;CURRENT AZIMUTH LOCATION
        STX       ELKEY      ;SAME FOR ELEVATION
        LDX       AZSIGN
        STX       AZKEYS     ;CURRENT AZ SIGN STATUS
;
        NOP                  ;FIX FOR THE "CLI" INSTR THAT FOLLOWS
        CLI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                     ;;
;;                          BEGIN STATE TABLES                         ;;
;;                                                                     ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
MSGA    LDX       #MSG12
         JSR       ASCDIS      ;DISPLAY "THE GA. TECH RFSS"
        LDA B     #20
        JSR       WAITE      ;WAIT FOR 2 SECONDS
        LDX       #MSG13
        JSR       ASCDIS     ;DISP "RADOME POSITIONER"
        LDA B     #10
        JSR       WAITE      ;WAIT FOR 1 SECOND
        LDX       #MSG14
        JSR       ASCDIS     ;DISPLAY " VERSION 1.5          "
MSGB    LDA B     #10
        JSR       WAITE
;
;         STATE ZERO
;       MAIN CONTROL LOOP
;
ST0     LDA A     KFLAG      ;IDLE STATE
        BPL       ST0A
        CLR       KFLAG      ;CLEARS "KEY PRESSED" FLAG
        LDA A     KEYENT     ;GETS KEYCODE OF KEY PRESSED
        LDX       #SP0       ;PUTS 0 STATE POINTER IN INDEX REG
        JSR       ADDCAL     ;JUMPS TO ROUTINE TO CALC NEXT STATE
        LDX       0,X
        JMP       0,X
        ;
        ; AZIMUTH MOTOR CONTROL LOOP
        ;
ST0A    JSR       SHAENC     ;READS AZ AND EL ANGLES
        LDA A     MFLAG
        BMI       ST0
        LDA A     AZKEYS
        CMP A     AZSIGN     ;SEE IF BOTH SIGNS EQUAL
```

```
        BEQ     STOX        ;BRANCH TO DO A BCDSUB IF SIGNS ARE SAME
        LDA A   AZKEY+1
        ADD A   AZBCD+1     ;FIND LSBYTE OF AZ MAGNITUDE DIFF
        DAA
        STA A   AZMAG+1
        LDA A   AZKEY
        ADC A   AZBCD       ;FIND MSBYTE OF AZ MAG DIFF
        DAA
        LDA B   AZMAG+1
        STA A   AZMAG       ;AZ MAG DIFF NOW IN A AND B REGS
        BRA     STOX2
STOX    LDA A   AZKEY
        LDA B   AZKEY+1
        LDX     #AZBCD      ;PUT ADDRESS OF BCD CURRENT LOC IN INDEX REG
        JSR     BCDSUB      ;JUMPS TO ROUTINE TO SUBTR BCD #"S
        STA A   AZMAG
        STA B   AZMAG+1
        ;
        ; ADDITION TO TIGHTEN CONTROL LOOP (AZ) TO .1 DEGREE (MOD 1.1)
        ;
STOX2   TST A               ;START <0.2 DEGREE TEST
        BNE     STOX1       ;BRANCHES TO <0.5 DEG TEST IF BCD WORD NOT <0.2
        CMP B   #15H        ;COMPARING T 0.15 DEG
        BHI     STOX1       ;BRANCHES TO <0.5 DEG TEST IF BCD WORD NOT <0.2
        LDA A   #0FFH       ;CURRENT POSITION IS LESS THAT 0.2 DEG
        STA A   SFLAGA      ;SETS AZ SPEED FLAG WITH CORRECT SPEED
        STA A   LSBSAZ      ;STOP AZ MOTOR WITH ZERO SPEED
        BRA     STOE
STOX1   TST A               ;START <0.5 DEG TST
        BNE     STOB        ;BRANCHES IF <0.5 DEG TEST IF BCD WORD NOT <0.5
        CMP B   #050H
        BHI     STOB        ;BRANCHES IF <0.5 DEG TEST IF BCD WORD NOT <0.5
        LDA A   #ETHSPD     ;SET SPEED TO EIGHTH SPEED
        STA A   SPEEDA      ;SET UP SPEED VARIABLE FOR USE LATER
        BRA     STOD
        ;
STOB    CMP A   #004H       ;TEST FOR <5 DEG
        BHI     STOC        ;BR TO <10 DEG IS <5 DEG TEST FAILS
        LDA A   #QUASPD     ;SET SPEED TO QUARTER SPEED
        STA A   SPEEDA      ;USED LATER
        BRA     STOD
STOC    LDA A   AZSPD       ;SET SPEED TO USER SELECTED SPEED (MOD 1.3)
        STA A   SPEEDA      ;USED LATER
STOD    LDA A   AZKEYS      ;DESTINATION NOT REACHED, CHECKS SIGNS
        CMP A   AZSIGN
        BEQ     SAMEAZ
DIFFAZ  CMP A   #02BH       ;DIFF SIGNS
        BEQ     B2
        JMP     LEFT
B2      JMP     RIGHT
SAMEAZ  CMP A   #02BH       ;SAME SIGN FIND WHICH ONE PLUS
        BEQ     YESAZ
NOAZ    TST     CARRY
        BMI     B3
        JMP     LEFT
B3      JMP     RIGHT
YESAZ   TST     CARRY
        BMI     B4
        JMP     RIGHT
B4      JMP     LEFT
        ;
        ;ELEVATION MOTOR CONTROL LOOP
        ;
STOE    LDA A   ELKEYS
        CMP A   ELSIGN
        BEQ     STOY        ;THIS CODE DUPLICATED FROM THE AZ CODE ABOVE
```

```
        LDA A     ELKEY+1
        ADD A     ELBCD+1
        DAA
        STA A     ELMAG+1
        LDA A     ELKEY
        ADC A     ELBCD
        DAA
        LDA B     ELMAG+1
        STA A     ELMAG
        BRA       STOY1
STOY    LDX       #ELBCD
        LDA A     ELKEY
        LDA B     ELKEY+1
        JSR       BCDSUB
        STA A     ELMAG
        STA B     ELMAG+1
        ;
        ; ADDED TO TIGHTEN ELEVATION CONTROL LOOP TO .1 DEGREE (MOD 1.1)
        ;
STOY1   TST A               ;THIS CODE ALSO DUPLICATED FROM AZ CODE
        BNE       STOZ
        CMP B     #10H
        BHI       STOZ
        LDA A     #0FFH
        STA A     SFLAGE
        STA A     LSBSEL
        JMP       CPFLAG
STOZ    TST A
        BNE       STOF
        CMP B     #050H
        BHI       STOF
        LDA A     #ETHSPD
        STA A     SPEEDE
        BRA       STOH
        ;
STOF    CMP A     #004H
        BHI       STOG
        LDA A     #QUASPD
        STA A     SPEEDE
        BRA       STOH
STOG    LDA A     ELSPD
        STA A     SPEEDE
STOH    LDA A     ELKEYS
        CMP A     ELSIGN
        BEQ       SAMEEL
DIFFEL  CMP A     #02BH
        BEQ       B5
        BRA       B7
B5      BRA       B8
SAMEEL  CMP A     #02BH
        BEQ       YESEL
NOEL    TST       CARRY
       BMI      B6
       BRA      B7
B6     BRA      B8
YESEL  TST      CARRY
       BMI      B7
B8     JMP      UP
B7     JMP      DOWN
       ;END STATE ZERO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                        ;;
;;                             BEGIN STATE 1                              ;;
;;                                                                        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
ST1      LDA A    #QUASPD   ;BEGIN STATE ONE, MANUAL DOWN
         LDA B    #0FFH
         JSR      MOTEL     ;JUMPS TO SUBROUTINE COMMAND ELEVATION MOTOR
ST1A     LDA A    #70H
         STA A    DDRA
         LDA A    DDRA
         CMP A    #77H      ;HAS DOWN KEY BEEN LET UP YET
         BNE      ST1B      ;DOWN KEY IS NOT BEING PRESSED NOW
         JSR      RESTO     ;RESTORE KEYBOARD BEFORE READING ANGLES (MOD 1.1)
         JSR      SHAENC    ;DOWN KEY NOT UP YET THEREFOR READ ANGLES
         BRA      ST1A
ST1B      CLR      LFLAGE    ; CLR LIM REACHED FLG.
         LDA A    #0FFH     ;TURN OFF MOTOR
         TAB                ;CLOCKWISE MOTION STILL SET
         JSR      MOTEL
         JSR      RESTO     ;RESTORE KEYBOARD PIA, AND BACK TO STATE ZERO
         JMP STO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                          BEGIN STATE 2                               ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST2      LDA A    #QUASPD   ;BEGIN STATE 2, LOAD A WITH SPEED (MOD 1.2)
         CLR B              ;LOAD B WITH DIRECTION (MOD 1.2)
         JSR      MOTEL
ST2A     LDA A    #70H
         STA A    DDRA
         LDA A    DDRA
         CMP A    #07BH     ;KEY RELEASED ?
         BNE      ST2B
         JSR      RESTO     ;RESTORE KEYBOARD BEF READING ANGLES
         JSR      SHAENC    ;UP KEY STILL BEING PRESSED. READ ANGLES
         BRA      ST2A
ST2B     CLR      LFLAGE    ;CLEAR LIMIT REACHED FLAG
         LDA A    #0FFH
         CLR B              ;C-CLOCKWISE MOTION SET
         JSR      MOTEL
         JSR      RESTO
         JMP      STO
         ;END STATE 2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                              BEGIN STATE 3                           ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST3      LDA A    #QUASPD   ;BEGIN STATE 3. LOAD A WITH SPEED (MOD 1.2)
         CLR B              ;LOAD B WITH DIRECTION (MOD 1.2)
         JSR      MOTAZ     ;JMP TO ROUTINE TO COMMAND AZIMUTH MOTOR
ST3A     LDA A    #0B0H
         STA A    DDRA
         LDA A    DDRA
         CMP A    #0B7H     ;HAS RIGHT KEY BEEN RELEASED?
         BNE      ST3B
         JSR      RESTO
         JSR      SHAENC
         BRA      ST3A
ST3B     CLR      LFLAGA    ;CLEAR LIMIT REACHED FLAG (MOD 1.1)
         LDA A    #0FFH     ;TURN AS MOTOR OFF
         CLR B
         JSR      MOTAZ
         JSR      RESTO     ;RESTORE KEYBOARD PIA. BACK TO STO
         JMP STO
         ;END STATE 3
```

```
;;                                                                   ;;
;;                           BEGIN STATE 4                           ;;
;;                                                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST4     LDA A   #QUASPD   ;BEGIN STATE 4. LOAD A=SPEED (MOD 1.2)
        LDA B   #0FFH     ;LOAD B WITH DIRECTION
        JSR     MOTAZ
ST4A    LDA A   #0B0H
        STA A   DDRA
        LDA A   DDRA
        CMP A   #0BBH     ;HAS LEFT KEY BEEN RELEASED?
        BNE     ST4B
        JSR     RESTO
        JSR     SHAENC
        BRA     ST4A
ST4B    CLR     LFLAGA    ;CLEAR LIMIT REACHED FLAG (MOD 1.1)
        LDA A   #0FFH
        TAB               ;CLOCKWISE MOTION
        JSR     MOTAZ
        JSR     RESTO
        JMP     ST0
        ;END STATE FOUR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                   ;;
;;                           BEGIN STATE 5                           ;;
;;                                                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST5     LDX     #MSG4     ;DISPLAYS "ERROR INVALID ENTRY"
        JSR     ASCDIS
        LDA B   #10
        JSR     WAITE
        JMP     ST0       ;BACK TO STATE 0
        ;END STATE 5
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                   ;;
;;                           BEGIN STATE 6                           ;;
;;                                                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST6     LDX     #MSG8
        JSR     ASCDIS    ;DISP "ANGLE TOO LARGE............"
        JMP     MSGB      ;WAIT 1 SEC THE RETURN TO CONTROL LOOP
        ;END STATE 6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                   ;;
;;                           BEGIN STATE 7                           ;;
;;                                                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST7     LDX     #MSG11
        JSR     ASCDIS    ;DISP "POSITIONER HALTED          "
        JMP     MSGB      ;WAIT 1 SEC THEN RETURN TO CONTROL LOOP
        ;END STATE 7
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                           BEGIN STATE 10                             ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST10    STA A   AZEL      ;REMEMBERS WHICH KEY PRESSED (SETAZ OR SETEL)
        LDX     #MSG2
        JSR     ASCDIS    ;DISP "ENTER AZIMUTH ANGLE"
        LDX     #DISEL+1
        STX     SAVEX     ;KEEPS TRACK OF WHERE THINGS ARE ON THE DISPLAY
```

```
ST10A    LDA A    KFLAG
         BPL      ST10A
         CLR      KFLAG      ;CLEARS KEYENTRY FLAG
         LDX      #MSG6
         JSR      ASCDIS     ;DISPLAY "AZIMUTH           ;
         CLR      ENTRYA
         CLR      ENTRYB     ;CLEAR BOTH REGS TO BE USED WHEN PACKING ENTRIES
         LDA A    KEYENT     ;GETS KEYENTRY
         LDX      #SP10
         JSR      ADDCAL
         LDX      0,X
         JMP      0,X        ;JUMPS TO NEXT STATE DETERMINED BY KEYENTRY IN A
         ;END STATE 10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                          ;;
;;                             BEGIN STATE 11                               ;;
;;                                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST11     STA A    AZEL       ;REMEMBERS KEY PRESSED (SETEL OR SETAZ)
         LDX      #MSG1
         JSR      ASCDIS     ;DISP "ENTR ELECATION ANGLE"
         LDX      #DISEL+1
         STX      SAVEX
ST11A    LDA A    KFLAG      ;WAITS FOR NEXT ENTRY
         BPL      ST11A
         CLR      KFLAG      ;CLEARS ENTRY FLAG
         LDX      #MSG7
         JSR      ASCDIS     ;DISPLAYS "ELEVATION         "
         CLR      ENTRYA
         CLR      ENTRYB
         LDA A    KEYENT     ;GETS KEYENTRY
         LDX      #SP11
         JSR      ADDCAL
         LDX      0,X
         JMP      0,X
         ;END STATE 11
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                          ;;
;;                            BEGIN STATE 12                                ;;
;;                                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST12     LDA B    #02BH      ;BEGIN STATE 12, PLUS SIGN AND MAGNITUDE
         LDX      SAVEX
         STA B    0,X        ;DIPLAYS PLUS SIGN
         INX                 ;INCREMENTS TRACKING POINTING
         STA B    TEMPS      ;REMEMBERS SIGN OF ENTRY
         LSR A               ;CONVERTS KEYCODE TO BCD CODE
         TAB
         JSR      PACK       ;ROUTINE TO PACK ENTRY
         ADD B    #030H      ;CONVERTS BCD CODE TO ASCII
         STA B    0,X        ;ECHOS KEYENTRY ON THE DISPLAY
         INX
         STX      SAVEX      ;REMEMBERS NEW VALUE OF TRACKING POINTER
ST12A    LDA A    KFLAG      ;WAIT FOR ANOTHER KEYENTRY
         BPL      ST12A
         CLR      KFLAG
         LDA A    KEYENT     ;GETS KEYENTRY
         LDX      #SP12
         JSR      ADDCAL
         LDX      0,X
         JMP      0,X
         ;END STATE 12
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
```

```
;;                              BEGIN STATE 13                                          ;;
;;                                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST13    LDA B   #02DH
        LDX     SAVEX
        STA B   0,X         ;DISPLAYS ENTERED MINUS SIGN
        INX                 ;INC TRACKING POINTER
        STX     SAVEX      ;SAVE TRACKING POINTER
        STA B   TEMPS
ST13A   LDA A   KFLAG
        BPL     ST13A
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP13
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 13
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                                    ;;
;;                              BEGIN STATE 14
;;                                                                                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST14    LSR A
        TAB                 ;ST14 DIPLAYS ENTERED NUM AFTER MINUS SIGN
        JSR     PACK
        ADD B   #030H
        LDX     SAVEX
        STA B   0,X         ;ECHO KEYENTRY
        INX                 ;INC TRACKING POINTER
        STX     SAVEX       ;KEEPS TRACK OF POINTER
ST14A   LDA A   KFLAG       ;WAITS FOR NEXT KEYENTRY
        BPL     ST14A
        CLR     KFLAG       ;CLEARS KEYENTRY FLAG
        LDA A   KEYENT      ;GETS KEYENTRY
        LDX     #SP14       ;LOADS INDEX REGISTER WITH STATE 14 POINTER
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 14
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                                    ;;
;;                              BEGIN STATE 15                                        ;;
;;                                                                                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST15    LSR A               ;CONVERTS KEYCODE TO BCD CODE
        TAB                 ;ST15 DIPLAYS SECOND NUM AFTER EITHER + OR -
        JSR     PACK
        ADD B   #030H
        LDX     SAVEX
        STA B   0,X
        INX
        STX     SAVEX
ST15A   LDA A   KFLAG
        BPL     ST15A       ;WAIT FOR NEXT KEYENTRY
        CLR     KFLAG       ;CLEARS KEYENTRY FLAG
        LDA A   KEYENT
        LDX     #SP15
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 15
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;                                                                      ;;
;;                          BEGIN STATE 16                              ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST16    LDA B   #02EH    ;ST16 DIPLAYS DECIMAL POINT (ENTERED)
        LDX     SAVEX
        STA B   0,X      ;ECHOS THE DECIMAL POINT
        INX
        STX     SAVEX
ST16A   LDA A   KFLAG
        BPL     ST16A    ;WAITS FOR NEXT KEYENTRY
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP16
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 16
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                          BEGIN STATE 17                              ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST17    LSR A            ;BEGIN STATE 17-DISPLAY LAST ENTERED NUM
        TAB
        JSR     PACK
        ADD B   #030H    ;CONV BCD CODE TO ASCII
        LDX     SAVEX
        STA B   0,X
        INX
        LDA A   #27H
        STA A   0,X      ;DISPLAYS DEGREE MARK AFTER LAST ENTERED NUM
        INX
        STX     SAVEX    ;INC AND SAVE TRACKING POINTER
        LDA A   AZEL
        CMP A   #01EH    ;TEST TO SEE IF SET EL WAS ENTERED
        BEQ     ST17A
        LDA A   TEMPS    ;STOES ENTERED DATA INTO APPROPRIATE AZ REGS
        STA A   AZKEYS
        LDA A   ENTRYA
        LDA B   ENTRYB
        JSR     TSTANG
        LDX     ENTRYA   ;TEST FOR ANGLE >40 DEG
        STX     AZKEY
        BRA     ST17C
ST17A   LDA A   TEMPS    ;STORES ENTERED DATA INTO APPROPRIATE EL REGS
        STA A   ELKEYS
        LDA A   ENTRYA
        LDA B   ENTRYB
        JSR     TSTANG   ;TST TO SEE IF ANGLE >40 DEG
        LDX     ENTRYA
        STX     ELKEY
        BRA     ST17C
ST17C   LDA A   KFLAG    ;WAITS FOR START KEY TO BE PRESSED
        BPL     ST17C
        CLR     KFLAG
        LDA A   KEYENT   ;GET ENTRY
        LDX     #SP17
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 17
```

```
;;                              BEGIN STATE 18                          ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST18    CLR     MFLAG    ;CLEARS MOTOR FLAG: ST18 GO TO CONTOL LOOP
        CLR     LFLAGE   ;CLEARS LIMIT REACHED FLAG
        CLR     LFLAGA
        JMP     STO
        ;END STATE 18
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                              BEGIN STATE 19                          ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST19    LDA A   #0FFH    ;ST19 DISABLES CONTROL LOOP (MOD 1.1)
        STA A   MFLAG    ;SETS MOTOR FLAG SO CNTRL LOOP DISABLED
        JSR     ALSTOP
        TST     LFLAGA   ;CHECK FOR AZ LIMIT REACHED
        BEQ     ST19A    ;GO IF NOT REACHED
        CLR A            ;AZ LIMIT REACHED
        LDA B   DFLAGA   ;GET CURR DIRECTION STATUS
        COM B            ;GET OPP DIRECTION
        JSR     MOTAZ
        LDA B   #10
        JSR     WAITE    ;ALLOW TIME FOR AZ MOTOR TO REPOSTITION GIMBAL
        LDA A   #0FFH
        LDA B   DFLAGA
        JSR     MOTAZ    ;STOP MOTOR, LIMIT NO LONGER EXCEEDED
        JMP     STO
ST19A   CLR     LFLAGA
        TST     LFLAGE   ;CHECK FOR EL LIMIT REACHED
        BEQ     ST19B    ;GO IF EL LIMIT NOT EXCEEDED
        CLR A            ;EL LIMIT REACHED
        LDA B   DFLAGE   ;GET CURR DIRECTION
        COM B            ;GET OPP DIR
        JSR     MOTEL
        LDA B   #10
        JSR     WAITE    ;WAIT FOR EL MOTOR TO FINISH MOVE
        LDA A   #0FFH
        LDA B   DFLAGE
        JSR     MOTEL    ;STOP EL MOTOR, LIMIT NO LONGER EXCEEDED
        CLR     LFLAGE
        JMP     STO
ST19B   JMP     ST7      ;DISPLAYS "POSITIONER HALTED",WAIT 1 SEC,STO
        ;END STATE 19
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                              BEGIN STATE 20
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST20    LDX     #MSG9
        JSR     ASCDIS   ;DISPLAY "ENTER PROG NUMBER"
        LDX     #DISEL+5
        STX     SAVEX
ST20A   LDA A   KFLAG
        BPL     ST20A    ;WAIT FOR KEYENTRY
        CLR     KFLAG
        CLR     ENTRYA
        CLR     ENTRYB
        LDX     #MSG10
        JSR     ASCDIS   ;DISPLAY "PROG   :ENTER      "
        LDA A   KEYENT
        LDX     #SP20
        JSR     ADDCAL
```

```
        LDX     0,X
        JMP     0,X
        ;END STATE 20
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                            BEGIN STATE 21                                  ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST21    LSR A   ;CONVERTS KEYCODE TO BCD CODE
        ADD A   #30H        ;CONVERT TO ASCII- ST21 DISPLAYS PROG NUM
        STA A   DISAZ+5     ;ECHOES THE ENTER PROG NUMBER
        STA A   PROGN       ;REMEMBERS THE PROGRAM NUM
        LDA B   #041H
        STA B   DISEL+3     ;DISPLAYS "A" AFTER ENTER
        STA B   PROGL
ST21A   LDA A   KFLAG
        BPL     ST21A
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP21
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 21
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                            BEGIN STATE 22                                  ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST22    LSR A               ;CONVERT KEYCODE TO BCD-ST22 IN #1 NUM OF 3 THAT
        TAB                 ;MAKE UP THE RASTER SCAN
        JSR     PACK
        ADD B   #030H
        LDX     SAVEX
        STA B   0,X
        INX
        STX     SAVEX
ST22A   LDA A   KFLAG
        BPL     ST22A
        CLR     KFLAG
        LDA A   KEYENT      ;GETS KEYENTRY
        LDX     #SP22
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 22
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                            BEGIN STATE 23                                  ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST23    LSR A               ;CONV KEYCODE TO BCD
        TAB                 ;ST23-DIS 2nd OF 3 NUM THAT MAKES UP RASTER SCAN
        JSR     PACK
        ADD B   #030H       ;CONV TO ASC
        LDX     SAVEX
        STA B   0,X
        INX
        STX     SAVEX
ST23A   LDA A   KFLAG
        BPL     ST23A
        CLR     KFLAG
        LDA A   KEYENT
```

```
        LDX     #SP23
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 23
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                          ;;
;;                            BEGIN STATE 24                                ;;
;;                                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST24    LDA B   #02EH
        LDX     SAVEX
        STA B   0,X         ;ST24-ECHOES DECIMAL POINT
        INX
        STX     SAVEX
ST24A   LDA A   KFLAG
        BPL     ST24A
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP24
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
        ;END STATE 24
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                          ;;
;;                            BEGIN STATE 25                                ;;
;;                                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ST25    LSR A       ;ST25-DIS LAST ENTERED NUM OF 3 FOR RASTER SCAN
        TAB
        JSR     PACK
        ADD B   #030H
        LDX     SAVEX
        STA B   0,X
        INX
        LDA B   #027H
        STA B   0,X
        LDX     #DISEL+5    ;ECHOES ENTERE NUM AND DEGREE MARK
        STX     SAVEX       ;STORES TRACKING POINTER
        LDA A   PROGL
        CMP A   #042H
        BEQ     ST25B       ;BRANCH IF PROGL IS "B"
        CMP A   #043H
        BEQ     ST25C       ;BRANCH IF PROGL IS "C"
        LDX     ENTRYA
        STX     PROGA
        LDA B   #10
        JSR     WAITE       ;1 SEC WAIT
        ;
        ;ADDITION TO ST25. CHECKS TO SEE IF PROG #4 IS
        ;BEING IMPLEMENTED (MOD 1.1)
        ;
        LDA B   PROGN       ;GET CURR PROGRAM COUNTER
        CMP B   #34H        ;COMPARES WITH ASCII "4"
        BEQ     ST25A       ;GO IF STATE=PROGRAM 4
        ;
        LDX     #MSG3
        JSR     ASCDIS      ;CLEARS DISPLAY
        LDX     #MSG10
        JSR     ASCDIS      ;DISPLAY "PROG  :ENTER"
        LDA A   PROGN
        STA A   DISAZ+5     ;DISPLAYS THE PROGRAM NUM
        LDA A   #042H
```

```
        STA A     DISEL+5    ;DISPLAYS A "B" AFTER ENTER
        STA A     PROGL      ;STORES PROGRAM LETTER
        JMP       ST21A
ST25B   LDX       ENTRYA
        STX       PROGB
        ;
        ;ADDITION TO ST25B. CKS SEE IF PROG3 BEING IMPL"D
        ;MOD (1.1)
        ;
        LDA B     PROGN
        CMP B     #33H       ;ASCII "3"
        BEQ       ST25A      ;GO IF STATE=PROG 3
        ;
        LDA B     #10
        JSR       WAITE
        LDX       #MSG3      ;CLEAR DIS
        JSR       ASCDIS
        LDX       #MSG10     ;DISPLAY "PROG  :ENTER"
        JSR       ASCDIS
        LDA A     PROGN
        STA A     DISAZ+5
        LDA A     #043H      ;DISPLAY "C"
        STA A     DISEL+3
        STA A     PROGL      ;STORES CURR PROG LETTER
        JMP       ST21A      ;JUMPS TO ENTER "C"
ST25C   LDX       ENTRYA
        STX       PROGC
ST25A   LDA A     KFLAG
        BPL       ST25A
        CLR       KFLAG
        LDA A     KEYENT
        LDX       #SP25
        JSR       ADDCAL
        LDX       0,X
        JMP       0,X
        ;END STATE 25
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                          BEGIN STATE 00                                    ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST00 (MOD 1.1) PROGRAM DISTRIBUTION STATE
;
ST00    LDA A     PROGN      ;GET PROG NUM
        CMP A     #31H
        BEQ       ST001      ;BR IF IN PROG #1
        CMP A     #32H
        BEQ       ST002      ;BR IF IN PROG #2
        CMP A     #33H
        BEQ       ST003      ;BR IF IN PROG #3
        JMP       ST29       ;GO TO PROG #4
ST001   JMP       ST26       ;GO TO PROG #1
ST002   JMP       ST27       ;GO TO PROG #2
ST003   JMP       ST28       ;GO TO PROG #3
        ;END STATE 00
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                          BEGIN STATE 26                                    ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST26 (MOD 1.2) PATTERN NUM ONE
;
ST26    LDX       #0         ;CLEAR X REG
        STX       MFLAG      ;CLEAR MOTOR FAG AND PROG FLAG
```

```
        LDX     PROGA       ;GETS TWO-BYTE RASTER PARAM "A"
        STX     AZKEY       ;ENTERS AZ PART OF FIRST POINT
        LDX     PROGC
        STX     ELKEY       ;SAVE TWO-BYTE ANSWER
        LDX     #2D2BH      ;GET ASCII "MINUS" AND "PLUS" VALS FOR SIGNS
        STX     AZKEYS      ;ENTER AZ,EL SIGN VALS
        LDX     #ST26A
        STX     STADDR      ;SAVES RETURN ADDRESS
        CLR     PROCNT      ;CLEARS PROGRAM STATE COUNTER
        DEC     PFLAG
        JMP     STO         ;SETS PROG FLAG,GO TO CNTRL LOOP,ANTICIPATE RETURN
ST26A   LDA B   #02BH
        STA B   AZKEYS      ;ENTER #2 POINT IN RASTER
        LDA B   ELSIGN      ;GET CURR EL SIGN VAL
        CMP B   #2BH        ;IS POSOTIONER ABOVE AZ AXIS
        BEQ     ST26A1      ;BR IF HAS RASTER IS NOT DONE
        LDA A   ELBCD       ;GET CURR EL POSITION
        LDA B   ELBCD+1
        LDX     #PROGC      ;X REG POINTS TO DESTINATION
        JSR     BCDSUB      ;(CURR EL POS)-(DEST)
        CMP A   #1          ;CK PROXIMITY IF POSITIONER TO TERMINATION POINT
        BHI     WEIRD       ;ABSOLUTE MAG > 1.XX DEG
        BRA     DONE        ;POSITIONER WITHIN TERMINATIN RANGE. STOP
WEIRD   TST     CARRY       ;CK IF PAST POINT
        BMI     ST26A1      ;RASTER STILL WORKING DO NOT STOP
DONE    LDX     #ST26D      ;DEST REACHED, STOP RASTER
        BRA     ST26A2
ST26A1  LDX     #ST26B      ;RASTER NOT FINISHED, GOTO NEXT POINT
ST26A2  STX     STADDR      ;SAVE RETURN ADDRESS
        INC     PROCNT      ;INC PROGRAM COUNTER
        JMP     STO         ;GO TO CONTROL LOOP, ANTICIPATE RETURN
ST26B   LDA B   ELSIGN      ;GET CURR EL SIGN
        CMP B   #2BH        ;HAS POS CROSSED AZ AXIS?
        BEQ     ST26B0      ;GO IF STILL ABOVE AXIS
        LDA A   ELKEY+1     ;CROSSED AXIS MUST COMPENSATE
        ADD A   PROGB+1     ;(CURR EL)+(INCREMENTAL ANGLE)
        DAA
        TAB                 ;SAVES LS INFO IN ACCB
        LDA A   ELKEY
        ADC A   PROGB
        DAA
        BRA     ST26B01     ;BR TO ENTER EL CO-ORDINATE
ST26B0  LDA A   ELKEY       ;CALC POINT #3
        LDA B   ELKEY+1     ;GET CURR EL POS
        LDX     #PROGB      ;GET ENTERED INC ANGLE
        JSR     BCDSUB
ST26B01 STA A   ELKEY       ;ENTER IS THIRD POINT
        STA B   ELKEY+1
        TST     CARRY       ;HAS POSITIONER CROSSED AZ AXIS?
        BEQ     ST26B1      ;BR IF NOT CROSSED AXIS
        LDA A   #2DH        ;ONCE CHANGED TO MINUS, WILL STAY MINUS
        STA A   ELKEYS      ;CHANGES EL SIGN TO MINUS AT CROSSING
ST26B1  LDA B   PROCNT      ;GETS CURRENT VAL OF PROG COUNTER
        CMP B   #3          ;IS PERIOD OF SCAN COMPLETE?
        BEQ     ST26B2      ;BR IF PERIOD NOT COMPLETE
        LDX     #ST26C      ;PERIOD OF SCAN WILL BE COMPLETE
        INC     PROCNT
        BRA     ST26B3
ST26B2  LDX     #ST26A      ;RET TO SECOND POINT
        CLR     PROCNT      ;CLEARS PROCNT,START SCAN OVER
ST26B3  STX     STADDR      ;SAVE RETURN ADDRESS
        JMP     STO         ;GO CONTRL LOOP, ANTICIPATE RETURN
ST26C   LDA B   #2DH        ;ENTER POINT 4
        STA B   AZKEYS      ;REVERSE SIGN OF AZ POSITION
        LDX     #ST26B      ;RETURN FOR POINT 5 INSTRUCTIONS
        STX     STADDR      ;SAVE RETURN ADDRESS
```

```
        INC     PROCNT    ;INC PROGRAM COUNTER
        JMP     ST0
ST26D   CLR     PFLAG     ;CLEAR PROGRAM FLAG
        JMP     ST7       ;DISPLAY"POSITIONER HALTED"
                          ;GO TO CONTROL LOOP AND DO NOT COME BACK
        ;END STATE 26
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;                           BEGIN STATE 27                                   ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST27-PATTERN #2
;
ST27    LDX     #0
        STX     MFLAG     ;THIS CODE SAME AS ABOVE
        LDX     PROGA
        STX     AZKEY
        LDX     PROGC
        STX     ELKEY
        LDX     #2D2BH
        STX     AZKEYS
        LDX     #ST27A
        STX     STADDR
        CLR     PROCNT
        DEC     PFLAG
        JMP     ST0
ST27A   LDA B   #02DH
        STA B   ELKEYS
        LDA B   AZSIGN
        CMP B   #2DH
        BEQ     ST27A1
        LDA A   AZBCD
        LDA B   AZBCD+1
        LDX     #PROGC
        JSR     BCDSUB
        CMP A   #1
        BHI     WEIRD1
        BRA     DONE1
WEIRD1  TST     CARRY
        BMI     ST27A1
DONE1   LDX     #ST27D
        BRA     ST27A2
ST27A1  LDX     #ST27B
ST27A2  STX     STADDR
        INC     PROCNT
        JMP     ST0
ST27B   LDA B   AZSIGN
        CMP B   #2DH
        BEQ     ST27B0
        LDA A   AZKEY+1
        ADD A   PROGB+1
        DAA
        TAB
        LDA A   AZKEY
        ADC A   PROGB
        DAA
        BRA     ST27B01
ST27B0  LDA A   AZKEY
        LDA B   AZKEY+1
      LDX     #PROGB    ;GET ENTERED INC COUNTER
      JSR     BCDSUB
ST27B01       STA A   AZKEY     ;ENTER IN THIRD POINT
        STA B   AZKEY+1
        TST     CARRY
        BEQ     ST27B1
```

```
         LDA B    #2BH
         STA B    AZKEYS
ST27B1   LDA B    PROCNT
         CMP B    #3
         BEQ      ST27B2
         LDX      #ST27C
         INC      PROCNT
         BRA      ST27B3
ST27B2   CLR      PROCNT
         LDX      #ST27A
ST27B3   STX      STADDR
         JMP      ST0
ST27C    LDA B    #2BH
         STA B    ELKEYS
         LDX      #ST27B
         STX      STADDR
         INC      PROCNT
         JMP      ST0
ST27D    CLR      PFLAG
         JMP      ST7
         ;END STATE 27
```

```
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                           ;;
;;                           BEGIN STATE 29                                  ;;
;;                                                                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST29-GENERATES ONE CIRCLE W/RADIUS "A" (MOD 1.2)
;
        ORG     2C00H
ST29    LDX     #0
        STX     PROANG     ;INIT ANGLE COUNTER
        STX     MFLAG     ;CLEAR PROGFLAG AND MOTFLAG
        LDA A   PROGA      ;GET ENTERED ANGLE "A"
        LDA B   PROGA+1
        JSR     DIVID100   ;DIVIDE ANGLE BY 100 (XX.XX=>00XX.)
        JSR     BCDBIN     ;CONVERT BCD TO BIN
        STA A   RADIUS     ;SAVE TWO BYTE RESULT
        STA B   RADIUS+1
        DEC     PFLAG
ST29A   LDX     PROANG     ;GET UPDATED ANGLE COUNTER
        CPX     #361       ;COMPARE PROANG TO 361 DEGREES
        BEQ     ST29B      ;BR IF PATTERN FINISHED
        LDX     #PROANG    ;SET X POINTER TO ANGLE POINTER
        JSR     TRGVALUE     ;GET SIN,COS
        LDX     #ST29A
        STX     STADDR
        LDX     PROANG     ;BEGIN UPDATING PROANG
        INX                ;INC PROANG BY 1 DEG
        STX     PROANG
        JMP     ST0
ST29B   CLR     PFLAG
        JMP     ST7
        ;END STATE 29
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                           ;;
;;                          BEGIN STATE 30                                   ;;
;;                                                                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST30-SET NEGATIVE ELEVATION LIMIT
;
ST30    LDX     #MSG16
        JSR     ASCDIS     ;DISPLAY "NEG EL LIMIT"
        LDX     #DISEL+6
        STX     SAVEX      ;SAVE DISPLAY TRACKING POINTER
        LDA A   NELLIM     ;GET CURRENT NEGATIVE ELEVATION LIMIT
        LDA B   NELLIM+1
        JSR     BCDDIS     ;DISPLAY CURRENT NEG EL LIM
```

```
        JSR     MOVED     ;FIX LAST ENTRY IN DECIMAL PT
        LDX     #ST30B
        STX     STADDR    ;STORE RETURN ADDRESS
ST30A   LDA A   KFLAG
        BPL     ST30A
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP30
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
ST30B   LDX     ENTRYA
        STX     NELLIM
        BRA     ST30A
        ;END STATE 30
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                          BEGIN STATE 31                              ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST31-SET POS EL LIMITS (MOD 1.1)
;
ST31    LDX     #MSG17
        JSR     ASCDIS    ;DISPLAY "POS EL LIMIT"
        LDX     #DISEL+6
        STX     SAVEX
        LDA A   PELLIM    ;GET POS EL LIMIT
        LDA B   PELLIM+1
        JSR     BCDDIS
        JSR     MOVED
        LDX     #ST31B
        STX     STADDR
ST31A   LDA A   KFLAG
        BPL     ST31A
        CLR     KFLAG
        LDA A   KEYENT
        LDX     #SP31
        JSR     ADDCAL
        LDX     0,X
        JMP     0,X
ST31B   LDX     ENTRYA
        STX     PELLIM
        BRA     ST31A
        ;END STATE 31
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                      ;;
;;                          BEGIN STATE 32                              ;;
;;                                                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST32-SET NEG AZIMUTH LIMIT
;
ST32    LDX     #MSG18
        JSR     ASCDIS    ;DISPLAY "NEG AZ LIMIT "
        LDX     #DISEL+6
        STX     SAVEX
        LDA A   NAZLIM    ;GET POS AZ LIMIT
        LDA B   NAZLIM+1
        JSR     BCDDIS
        JSR     MOVED
        LDX     #ST32B
        STX     STADDR
ST32A   LDA A   KFLAG
        BPL     ST32A
        CLR     KFLAG
```

```
        LDA A    KEYENT
        LDX      #SP32
        JSR      ADDCAL
        LDX      0,X
        JMP      0,X
ST32B   LDX      ENTRYA
        STX      NAZLIM      ;UPDATE NEGATIVE AZIMUTH LIMIT
        BRA      ST32A
        ;END STATE 32
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                       ;;
;;                           BEGIN STATE 33                              ;;
;;                                                                       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST33-SET POSITIVE AZIMUTH LIMIT (MOD 1.1)
;
ST33    LDX      #MSG19
        JSR      ASCDIS      ;DISPLAY "POS AZ LIMIT"
        LDX      #DISEL+6
        STX      SAVEX
        LDA A    PAZLIM
        LDA B    PAZLIM+1
        JSR      BCDDIS      ;DISPLAY CURRENT POS AZ LIMIT
        JSR      MOVED
        LDX      #ST33B
        STX      STADDR
ST33A   LDA A    KFLAG
        BPL      ST33A
        CLR      KFLAG
        LDA A    KEYENT
        LDX      #SP33
        JSR      ADDCAL
        LDX      0,X
        JMP      0,X
ST33B   LDX      ENTRYA
        STX      PAZLIM
        BRA      ST33A
        ;END STATE 33
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                           BEGIN STATE 34                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST34-INPUT NUMBERS FOR SETTING LIMITS--1ST
;
ST34    LSR A                ;BEGIN INPUT,CONVERT KEYCODE TO BCDCODE
        TAB
        JSR      PACK        ;PACKS BCD INPUT INTO PACKED BCD FORM
        ADD B    #30H        ;CONVERT TO ASCII #
        LDX      SAVEX
        STA B    0,X
        INX                  ;INC TRACKING POINTER
        STX      SAVEX
ST34A   LDA A    KFLAG
        BPL      ST34A
        CLR      KFLAG
        LDA A    KEYENT
        LDX      #SP34
        JSR      ADDCAL
        LDX      0,X
        JMP      0,X
        ;END STATE 34
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          BEGIN STATE 35                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
        ;ST35-INPUT BCD NUMBER --2ND
;
ST35    LSR A     ;CONVERT KEYCODE TO BCD CODE
        TAB
        JSR       PACK
        ADD B     #30H
        LDX       SAVEX
        STA B     0,X
        INX
        STX       SAVEX
ST35A   LDA A     KFLAG
        BPL       ST35A
        CLR       KFLAG
        LDA A     KEYENT
        LDX       #SP35
        JSR       ADDCAL
        LDX       0,X
        JMP       0,X
        ;END STATE 35
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          BEGIN STATE 36                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST36-INPUT DECIMAL POINT
;
ST36    LDA B     #2EH      ;CONVERT INPUT TO BCD CODE
        LDX       SAVEX
        STA B     0,X
        INX
        STX       SAVEX
ST36A   LDA A     KFLAG
        BPL       ST36A
        CLR       KFLAG
        LDA A     KEYENT
        LDX       #SP36
        JSR       ADDCAL
        LDX       0,X
        JMP       0,X
        ;END STATE 36
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          BEGIN STATE 37                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST37-INPUT LAST BCD CHARACTER--3RD
;
ST37    LSR A               ;CONVERT INPUT KEYCODE TO BCD CODE
        TAB
        JSR       PACK
        ADD B     #30H
        LDX       SAVEX
        STA B     0,X
ST37A   LDA A     ENTRYA
        LDA B     ENTRYB
        JSR       TSTANG    ;TEST FOR ANGLE LIMIT CONDITION
        LDX       STADDR
        JMP       0,X
        ;END STATE 37
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          BEGIN STATE 38                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST38-SET AZ GIMBAL SPEED
;
ST38    LDY       #MSG21
        JSR       ASCDIS    ;DISPLAY 'AZ GIMBAL SPEED"
        LDX       [illegible]
```

```
        STX       SAVEX
        LDA A     AZSPEED    ;GET LAST USER SELECTED GIMBAL SPEED
        STA A     0,X
        LDX       #ST38B
        STX       STADDR     ;SAVES RETURN ADDRESS
ST38A   LDA A     KFLAG
        BPL       ST38A
        CLR       KFLAG
        LDA A     KEYENT
        LDX       #SP38
        JSR       ADDCAL
        LDX       0,X
        JMP       0,X
ST38B   LDA B     SPEED      ;GET USER SET GIM SPEED (ASCII)
        STA B     AZSPEED    ;SAVE AZ GIM SPEED
        LDA B     GIMSPEED   ;GT HEX FORM OF GIM SPEED
        STA B     AZSPD      ;SAVE HEX FORM OF GIMBAL SPEED
        BRA       ST38A      ;GO BACK TO WAIT FOR ANOTHER KEYENTRY
        ;END STATE 38
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                           BEGIN STATE 39                             ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST39-SET EL GIMBAL SPEED (MOD 1.3)
;
ST39    LDX       #MSG22
        JSR       ASCDIS     ;DISPLAY "EL GIMBAL SPEED"
        LDX       #DISEL+8
        STX       SAVEX
        LDA A     ELSPEED    ;GET LAST USER SELECTED GIMBAL SPEED
        STA A     0,X
        LDX       #ST39B
        STX       STADDR
ST39A   LDA A     KFLAG
        BPL       ST39A
        CLR       KFLAG
        LDA A     KEYENT
        LDX       #SP39
        JSR       ADDCAL
        LDX       0,X
        JMP       0,X
ST39B   LDA B     SPEED      ;GET USER SEL GIM SPD (ASCII)
        STA B     ELSPEED    ;SAVE EL GIM SPEED
        LDA B     GIMSPEED   ;GET HEX FORM OF GIM SPD
        STA B     ELSPD      ;SAVE HEX FORM OF GIM SPD
        BRA       ST39A
        ;END STATE 39
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                       BEGIN STATE 40                                     ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        ;ST40-INPUT SINGLE DIGIT GIMBAL SPEED (MOD 1.3)
;
ST40    LSR A                ;CONVERT KEYCODE TO 0-9
        TAB
        ADD B     #30H
        LDX       SAVEX
        STA B     0,X
        STA B     SPEED
        LDX       #SPDTB
        JSR       ADDCAL
        LDA B     0,X
        STA B     GIMSPEED
        LDX       STADDR
        JMP       0,X
        ;END STATE 40
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  FPTBCD- SUBROUTINE                                                ;;
;;      CONVERSION OF 32 BIT FLOATING POINT NUMBER INTO A 32          ;;
;;      BIT PACKED BCD. ROUTINE ASSUMES 32 FLT PT NUM="FPT32"         ;;
;;      ENTRY: X-REG POINTS TO LOCATION OF RESULT                     ;;
;;      EXIT : BCD RESULT SPECIFIED BY X REG, REG A,B DESTROYED       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
FPTBCD  STX     SAVEX1      ;SAVES RESULT POINTER
        CLR     INTREG      ;INITIALIZES INTEGER REGISTERS
        CLR     INTREG+1
        CLR     O,X         ;INIT BCD SIGN VAL
        LDX     #FPT32      ;SET POINTER TO 32 BIT NUM
        LDA A   O,X         ;GET EXP PART OF 32 BIT NUM
        BEQ     FPT7        ;DONT SHIFT IF EXP=O
        BPL     FPT1        ;BR IF 32 BIT NUM >O
        LDX     SAVEX1      ;IT IS NEG, FIX BCD SIGN BYTE TO BE NEG
        DEC     O,X         ;SIGN BYTE NEGATIVE
        LDX     #FPT32      ;RESET DATA POINTER
FPT1    ASL A               ;CONVERTS 7 BIT 2S COMP TO 8 BIT COMP
        ASR A
        BPL     FPT4        ;BR TO POS SHIFT ROUT IF 32 BIT # POS
        LDA B   #17         ;INIT BIT COUNTER (MOD 1.4)
FPT2    LSR     1,X         ;MOVES BINARY POINT OVER TO LEFT
        ROR     2,X         ;(MOD 1.4)
        ROR     3,X         ;(MOD 1.4)
        DEC B               ;DEC BIT COUNTER
        BNE     FPT3        ;BR IF UNDERFLOW DOES NOT OCCUR
        BRA     FPT7        ;BR TO END OF ROUTINE
FPT3    INC A               ;ADJUSTS THE EXP
        BNE     FPT2        ;BR THRU LOOP UNLESS BIN POINT ADJUSTED
        BRA     FPT7        ;BR TO CONVERSION PART OF ROUTINE
FPT4    LDA B   #17         ;INIT BIT COUNTER (MOD 1.4)
FPT5    ASL     3,X         ;BEGIN POS SHIFT ROUTINE
        ROL     2,X         ;ROTATE THIRD BYTE OF 32 BIT NUM
        ROL     1,X         ;ROTATE SECND BYTE OF 32 BIT NUM
        ROL     INTREG+1    ;ROTATES 32 BIT NUM INTO INTEGER REG
        ROL     INTREG
        DEC B               ;DEC BIT COUNTER
        BNE     FPT6        ;BR IF OVERFLOW HAS NOT OCCURED
        SEV                 ;SETS OVERFLOW BIT TO DENOTE AN ERROR
        BRA     FPT8        ;BR TO END OF ROUTINE
FPT6    DEC A               ;ADJUST EXP
        BNE     FPT5        ;BR THRU LOOP UNLESS BIN PT ADJSTED
FPT7    LDA A   1,X         ;GET BIN FRACTION IN ACCA
        LDA B   2,X         ;(MOD 1.4)
        PSH A               ;SAVE A
        PSH B               ;SAVE B (MOD 1.4)
        LDX     SAVEX1      ;SETS DATA POINTER
        INX
        LDA A   INTREG      ;RETRIEVES THE BIN INTEGER
        LDA B   INTREG+1
        BSR     BINBCDED    ;CONVERT INTEGER PART
        LDX     SAVEX1      ;SETS DATA POINTER
        INX
        INX
        INX
        PUL B               ;(MOD 1.4)
        PUL A
        BSR     BINFPT      ;CONVERT FRACTIONAL PART
FPT8    RTS                 ;RETURN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "BINBCDED"-SUBROUTINE                                                    ;;
;;      BINARY TO PACKED BCD CONVERSION ROUTINE                              ;;
;;      LOAD ACCA,ACCB WITH A 16 BIT BIN NUM                                 ;;
;;      LOAD INDEX REG WITH ADDRESS OF MS BYTE OF WHERE PUT PACKED BCD ANS   ;;
```

```
;;      RETURNS WITH PACKED INFO IN SPEC LOCATION AND IN ACCA,ACCB         ;;
;;      THE LEAST 4 MS DECIMAL VALS WILL BE CONTAINED IN THE PACKED BCD ANS;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
BINBCDED STX        SAVEX      ;SAVE DATA POINTER
         LDX        #K10K      ;INITS X-REG FOR 1ST BCD CONV CONST
         CLR        ENTRY1
         CLR        ENTRY2
ZVDEC1   CLR        SAVEA      ;CLEAR BCD CONVERSION COUNTER
ZVDEC2   SUB B      1,X
         SBC A      0,X
         BCS        ZVDEC5     ;BR IF SUB PRODUCES OVERFLOW
         INC        SAVEA      ;DEC CHAR BEING BUILT, INC SAVEA
         BRA        ZVDEC2
ZVDEC5   ADD B      1,X
         ADC A      0,X        ;RESTORES PARTIAL RESULT UPON OVERFLOW
         PSH        A                    ;SAVE ACCA
         LDA A      SAVEA      ;GETS BCD CONVERSION COUNTER
         BSR        PACKED     ;PACKS NEWLY FORMED BCD CHARACTER
         PUL A                 ;RESTORES ACCA TO FORMER VALUE
         INX
         INX
         CPX        #K10K+10   ;TESTS TO SEE IF LAST CONSTANT HAS BEEN USED
         BNE        ZVDEC1
         LDA A      ENTRY1     ;LAST CHARACTER HAS BEEN REACHED
         LDA B      ENTRY2
         LDX        SAVEX
         STA A      0,X
         STA B      1,X        ;SAVES 16
         RTS
         ;END BINBCDED SUBROUTINE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    "PACKED" SUBROUTINE                                                  ;;
;;       PACKS BINARY NUMBER INTO BCD FORM                                 ;;
;;       ACCA SHOULD CONTAIN THE UNPACKED BCD FORM                         ;;
;;       ROUTINE DESTROYS CONTENTS OF ACCA                                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
         ;
;
PACKED   ASL        ENTRY2     ;ONE BIT LEFT SHIFT WITH ZERO FIL
         ROL        ENTRY1
         ASL        ENTRY2
         ROL        ENTRY1
         ASL        ENTRY2
         ROL        ENTRY1
         ASL        ENTRY2
         ROL        ENTRY1     ;SHIFTS 16 BIT BINARY INFO OVER ONE CHAR
         ADD A      ENTRY2
         STA A      ENTRY2     ;ENTRY2 FORM="X0",PACKS ANOTHER UNPACKED FORM
         RTS                   ;RETURN FORM SUBROUTINE
         ;END PACKED
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   "BINFPT" SUBROUTINE                                                   ""
;;      CONVERSION OF FRACTIONAL PART OF BINARY NUM TO PACKED BCD          ;;
;;      LOAD FRACTIONAL PART IN ACCA BEFORE EXECUTING                      ;;
;;      ACCB IS USED IN CALCULATION                                        ;;
;;      ROUTINE EXITS WITH BCD ANSWER (4 DEC PLACES) IN ACCA,ACCB          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
BINFPT   STX        SAVEX      ;SAVES DATA POINTER
         STA A      SAVEA      ;SAVE FRACT PART
         STA B      SAVEB      ;(MOD 1.4)
         LDA B      #16        ;(MOD 1.4)
         STA B      SAVEC      ;SAVE BIT COUNTER (MOD 1.4)
         LDX        #CONST     ;SET POINTER IN ACCX AT FIRST BYTE OF CONSTANTS
         CLR A
```

```
        CLR B
BIN1    STA A       TEMPA      ;SAVE ACCA TEMPORARILY
        ASL         SAVEB      ;LOOK AT NEXT BIT
        ROL         SAVEA
        BCC         BIN2       ;BR PAST LOOP IF C=0
        TBA                    ;OUT ACCB INTO ACCA
        ADD A       1,X        ;C=1,ADD IN CONSTANT
        DAA
        TAB
        LDA A       TEMPA      ;RETRIEVE ACCA FROM TEMP STORAGE
        ADC A       0,X
        DAA
BIN2    INX                    ;INC ACCX TO NEXT CONSTANT
        INX
        DEC         SAVEC      ;DEC BIT COUNTER
        BNE         BIN1       ;BR THRU LOOP UNTIL 8 BITS ARE SHIFTED
        LDX         SAVEX      ;RETREIVE DATA POINTER
        STA A       0,X        ;SAVE 16 BIT PACKED BCD CHARACTE
        STA B       1,X
        RTS                    ;RET FROM SUBROUTINE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "DIVID100" SUBROUTINE                                              ;;
;;      DIVIDES BCD VAL BY 100                                         ;;
;;      ENTRY: ACCA,ACCB CONTAIN 16 BIT BCD NUMBER                     ;;
;;      EXIT : ACCA,ACCB CONTAIN 16 BIT BCD RESULT                     ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DIVID100 TAB                   ;THROWS AWAY THE FRACTIONAL PART
        CLR A                  ;ACCA,ACCB="00XX"
        RTS                    ;RETURN FROM SUBROUTINE
        ;END DIVID100
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "CHSIGN" SUBROUTINE                                                ;;
;;      CHANGES THE SIGN OF A TWO BYTE COORD FROM ONE STATE            ;;
;;      TO THE OTHER                                                   ;;
;;      ENTRY: ACCB CONTAINS SIGN INFO,X REG POINTS TO LOCATION OF     ;;
;;             SIGN INFO                                               ;;
;;      EXIT : ORIGINAL SIGN INFO AUTOMATICALLY CHANGED,ACCB DESTROYED ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;
CHSIGN  CMP B       #2BH       ;FIND OUT SIGN
        BEQ         CHSIGN1    ;BR IF PLUS
        LDA B       #2BH       ;SIGN IS MINUS
        BRA         CHSIGN2
CHSIGN1 LDA B       #2DH       ;CH PLUS TO MINUS
CHSIGN2 STA B       0,X        ;UPDATE SIGN VAL
        RTS                    ;RETURN
        ;END CHSIGN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "CONSIGN" SUBROUTINE                                               ;;
;;     GETS SIGN OF 32 BIT FP # AND FINDS ASCII EQUIV                  ;;
;;     ENTRY: ACCB CONTAINS SIGN PORTION OF FP #                       ""
;;            X REG CONTAINS LOCATION OF RESULTING SIGN                ;;
;;     EXIT : APPROPRIATE SIGN IS LOCATED, ACCB DESTROYED              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
CONSIGN BMI         CONSIGN1   ;BR IF NEG
        LDA B       #2BH       ;SIGN IS PLUS ASCII = 2BH
        BRA         CONSIGN2
CONSIGN1 LDA B      #2DH       ;SIGN IS MINUS ASCII IS 2DH
CONSIGN2 STA B      0,X        ;UPDATE SIGN VAL
        RTS                    ;RETURN
        ;END CONSIGN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "TRGVALUE" SUBROUTINE         - SIN,COS OF BINARY ANGLE            ;;
;;      ENTRY:  ACCB=# BYTES, X-REG POINTS TO ADDR OF BIN ANGLE        ;;
```

```
;;      EXIT :  2 BYTE VAL AZKEY AND ELKEY ARE REPLACED, ACCB DESTROYED  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
TRGVALUE LDA B      #2H        ;ACCB=#BYTE TO BE PUT AT TOS
         BSR        PUSH       ;PUSH ANGLE ONTO APU STACK
         BSR        FLTS       ;CONVERT ANG TO 32 BIT FLT PT FORM
         LDX        #RADIANS   ;X-REG POINTS TO PI/180 CONST
         LDA B      #4H
         BSR        PUSH
         BSR        FMUL       ;BIN ANG * PI/180 = ANGLE IN RADIANS RESULT TOS
         BSR        PTOF       ;DUP BIN ANGLE AT NOS
         BSR        COS        ;TAKE COS OF ANGLE
         LDX        #RADIUS    ;X-REG POINT TO  RADIUS 16 BIT FIXED POINT
         LDA B      #2H
         BSR        PUSH
         BSR        FLTS       ;CONVERT RADIUS TO 32 BIT FL PT #
         BSR        PTOF       ;PUSH 32 BIT TOS TO NOS
         BSR        POPF       ;32 BIT APU STACK POP
         BSR        FMUL       ;RADIUS*COS(ANG)= ELEVATION
         LDX        #FPT32
         LDA B      #4H
         BSR        PULL       ;PULL ELEVATION FROM APU STACK
         LDX        #ELRESULT  ;X POINTS TO LOCATION OF RESULT
         JSR        FPTBCD     ;CONVERT 32 BIT FP TO 4 BYTE BCD RESULT
         BSR        SIN
         BSR        FMUL       ;RADIUS*SIN(ANG)=AZ
         LDX        #FPT32
         LDA B      #4H
         BSR        PULL
         LDX        #AZRESULT  ;X-REG POINTS TO RESULT
         JSR        FPTBCD     ;CONVERT 32 BIT FP TO 4 BYTE BCD RESULT
         LDX        #ELKEYS
         LDA B      ELRESULT   ;GET 1ST BYTE OF RESULT
         BSR        CONSIGN    ;CONVERTS FP SIGN TO ASCII VAL
         LDX        #AZKEYS
         LDA B      AZRESULT
         BSR        CONSIGN
         LDX        #ELRESULT
         LDA A      2,X        ;GET MSBYTE OF MAG VAL OF FPT
         LDA B      3,X        ;GET LSBYTE OF MAG VAL OF FPT
         STA A      ELKEY      ;UPDATE EL COORD
         STA B      ELKEY+1
         LDX        #AZRESULT
         LDA A      2,X        ;GET MAG VAL OF FPT
         LDA B      3,X
         STA A      AZKEY
         STA B      AZKEY+1
         RTS                   ;RETURN
         ;END TRGVALUE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; "PUSH" SUBROUTINE  - MOVES B BYTES OF DATA ONTO APU STACK        ;;
;;    ENTRY: ACCB CONTAINS # BYTES TO PUSH ONTO STACK               ;;
;;           ACCX CONTAINS ADDRESS OF MSBYTE OF DATA TO PUSH        ;;
;;    EXIT : DATA WILL BE PLACED ON APU STACK SUCH THAT MSB OF      ;;
;;           WILL BE TOS.  X-REG DESTROYED , ACCB DESROYED          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PUSH     PSH        B          ;SAVES THE NUMBER OF BYTES
PUSH1    INX
         DEC B
         BNE        PUSH1              ;BR THRU TILL LAST BYTE REACHED
         STX        TEMPX
         PUL B                 ;RETRIEVES # BYTES TO BE PUSHED
PUSH2    DEX                   ;ACCESS NEXT ADD TO PUT ONTO APU STACK
         LDA A      0,X
         STA A      APUDATA    ;ENTER CURR BYTE OF DATA ONTO APU STACK
```

```
         DEC B
         BNE       PUSH2
         LDX       TEMPX
         RTS
         ;END PUSH
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; "FLTS" SUBROUTINE
;;   CONVERT 16 BIT FIX PT TO 32 BIT FP
;;   ENTRY: 16 FIXED # ON TOS
;;   EXIT : WHEN APU FINISHED
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
FLTS     LDA B     #1DH      ;LOADS IN FLOAT OPCODE
         BRA       TSTEND    ;WAIT TILL APU FINE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  "PTOF" SUBROUTINE
;;    DUP TOS AT NOS
;;    ENTRY: 32 BIT FP # AT TOS
;;    EXIT : WHEN APU FINISHED
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PTOF     LDA B     #17H
         BRA       TSTEND
;
;   COS SUBROUTINE
;     32 BIT FLOATING COSINE
;     ENTRY: 32 BIT FP AT TOS
;     EXIT : WHEN APU DONE
;
COS      LDA B     #3H       ;LOAD IN COS COMMAND
         BRA       TSTEND
;
; "POPF" SUBROUTINE - POP NOS INTO TOS
;   ENTRY: 32 BIT FP # AT TOS
;   EXIT : WHEN APU DONE
;
POPF     LDA B     #18H
         BRA       TSTEND
;
; "FMUL" SUBROUTINE - 32 BIT FLOATING POINT MULTIPLIER
;   ENTRY: 2 #"S ON TOS AND NOS
;   EXIT : WHEN APU DONE
;
FMUL     LDA B     #12H
         BRA       TSTEND
;
; "PULL" SUB - REMOVES B BYTES OF DATA FROM THE  STACK
;  ENTRY: ACCB = # BYTES TO BE PULLED
;         ACCX = ADDRESS TO WHERE TOS TO BE PLACED
;  EXIT : WHEN DONE
;
PULL     LDA A     APUDATA
         STA A     0,X
         INX
         DEC B
         BNE       PULL
         RTS
;
; "SIN" SUB - 32 BIT FP SIN
;    ENTRY: 32 BIT FP AT TOS
;    EXIT : WHEN DONE
;
SIN      LDA B     #2H
         BRA       TSTEND
;
; "XCHF" SUB - EXCH 32 BIT OPERANDS TOS AND NOS
```

```
;  ENTRY: BOTH #"S MUST BE ON APU STACK BEFOR XEQ
;  EXIT : WHEN DONE
;
XCHF     LDA B     #19H
         BRA       TSTEND
;
; "FADD" SUB - 32 BIT FLOATING POINT ADDITION
;  ENTRY: BOTH #"S ON STACK
;
FADD     LDA B     #10H
         BRA TSTEND
;
; "FSUB" SUBROUTINE - 32 BIT FP SUBRACT
;  ENTRY: BOTH #"S ON TOS AND NOS
;  EXIT : RESULT ON TOS
FSUB     LDA B     #11H
         BRA       TSTEND
;
; "FDIV" SUB- DIVIDE 2 32 BIT FP #"S
;   NOS/TOS
;   ENTRY: 2 32-BIT #"S ON NOS AND TOS
;
FDIV     LDA B     #13H
         BRA       TSTEND
;
; "TSTEND" SUBROUTINE - LOOPS UNTIL ENDFLAG FROM APU LOW (PA7 OF PIA2)
;   ENTRY: ISSUE COMMAND TO APU
;   EXIT : WHEN APU DONE
;
TSTEND   STA B     APUSTAT   ;ISSUES COMM TO APU
TSTEND1  LDA A     DDRA3     ;CK ENDFLAG
         BPL       TSTEND1   ;LOOP TILL LOW
         LDA A     APUSTAT   ;CLEAR FLAG
         RTS
;
;  "RESTO" SUBROUTINE - RETURNS THE KEYBOARD TO AN INITITIALIZED STATE
;          SO THAT ANY KEY PRESSED WILL GENERATE AN INTERRUPT
;
RESTO    LDA A     #0FH
         STA A     DDRA      ;RESTORE THE ROWS OF KBD FOR NXT KEY PUSHED
         LDA A     DDRA      ;CLEAR IRQ BITS IN CRA
         LDA A     #0FFH
         STA A     MFLAG     ;DIS CONTROL LOOP
         RTS
;
;  "MOVED" SUBROUTINE - INSERTS A DECIMAL POINT IN FRONT OF LAST
;          BCD CHARACTER (USED IN ST30-33)
;
MOVED    LDA A     DISEL+8
         LDA B     #2EH
         STA B     DISEL+8
         STA A     DISEL+9
         RTS
;
;  "CPFLAG" SUBROUTINE - (MOD 1.1)
;
CPFLAG   LDA A     SFLAGA
         CMP A     #0FFH
         BNE       CP1
         TST       PFLAG     ;LOOKS AT PROGRAM FLAG
         BEQ       CP1       ;BR TO CNTRL LOOP IF NOT IN PROGRAMMED SEQ
         LDX       STADDR    ;YES,A PROGRAMMED SEQ CURRENTLY IN OPERATION
         JMP       0,X       ;JMPS BACK TO THE PROGRAMMED CONTROL LOOP
CP1      JMP       ST0       ;PROGRAM FLAG CLEARED,GOTO CNTRL LP
;
;  "TSTANG" SUBROUTINE
```

```
;              COMPARES THE TWO ACCX"S WITH THE CONTENTS OF THE
;              INDEX REGISTER. RETURNS FROM SUB IF CONTENTS OF
;              INDEX REGISTER ARE > THE CONTENTS OF THE ACCX"S
;              BRANCHES TO ERROR 6 IF NOT
;
TSTANG   LDX        #LIMIT
         JSR        BCDSUB
         TST        CARRY
         BMI        NOPE
         JMP        ST6        ;BR IF ENTERED ANGLE EXCEEDS LIMIT OF +/- 40.0
NOPE     RTS
;
;   "BCDBIN" SUBROUTINE
;              CONVERTS FOUR BINARY CODED DECIMAL DIGIT
;              TO A BINARY EQUIVALENT. THE BCD DIGITS ARE PACKED
;              TWO PER BYTE. THE BINARY RESULT OCCUPIES TWO BYTES
;              THE BCD DIGITS ARE LOADED INTO THE ACCA AND ACCB
;              (MSD-ACCA) AND THE BCDBIN SUBROUTINE IS CALLED.
;              THE ROUTINE EXITS WITH THE BINARY RESULT IN ACCA
;              AND ACCB (MOD 1.1)
;
BCDBIN   STA A      SAVE1      ;SAVE 2 BCD VALS
         CLR        BINUPR
         TBA
         AND B      #0FH       ;SAVE ONLY LS BCD VAL
         LSR        A
         LSR        A
         LSR        A
         LSR        A
TENLP    BEQ        DOHUND     ;GO DOHUND WHEN TEN IS ZERO
         ADD B      #10        ;ADD 10 TO BINARY TOTAL
         DEC A                 ;DEC TENS DIGIT AND
         BRA        TENLP      ;REPEAT UNTIL 0
DOHUND   CLC
         LDA A      SAVE1      ;GET HUN IN THOU DIGIT
         AND A      #0FH       ;SAVE ONLY HUN DIGIT
HUNLP    BEQ        DOTHOU     ;DO THOU IF HUN IS 0
         ADD B      #100       ;ADD 100 TO BINARY VAL
         BCC        HUNOO
         INC        BINUPR     ;ADD 256 TO BINARY UPPER VAL
HUNOO    DEC A                 ;DEC HUN DIGIT ONE
         BRA        HUNLP      ;REPEAT TIL 0
DOTHOU   LDA A      SAVE1      ;GET THOU DIGIT
         LSR A
         LSR A
         LSR A
         LSR A
         STA A      SAVE1      ;SAVE THOU DIGIT
         BNE        THOUOO     ;BR IF THOU DIGIT = 0
         LDA A      BINUPR     ;GET BIN UPPER VAL
         BRA        XITBIN
THOUOO   LDA A      BINUPR     ;GET BIN UPPER VALUE
THOULP   CLC                   ;RESET CARRY
         ADD B      #232       ;ADD 232 TO BINARY LOWER
         ADC A      #3H        ;ADD 768 TO BINARY UPPER
         DEC        SAVE1      ;DEC THOU DIGIT
         BNE        THOULP     ;REPEAT TILL THOU DIGIT = 0
XITBIN   RTS
;
;   "PACK" SUBROUTINE
;           PACKS BINARY #"S INTO BCD FORM
;           ACCA=UNPACKED BCD FORM
;           DESTROYS ACCA
;
PACK     ADD A      ENTRYB     ;ENTRYB LOOKS LIKE "XO"
         STA A      ENTRYB     ;PACKS IN ANOTHER BCD FORM
```

```
        ASL       ENTRYB
        ROL       ENTRYA
        ASL       ENTRYB
        ROL       ENTRYA
        ASL       ENTRYB
        ROL       ENTRYA
        ASL       ENTRYB
        ROL       ENTRYA
        RTS
;
;   "ALSTOP" SUBROUTINE
;           ROUTINE THAT STOPS BOTH MOTORS FOR EXCEEDING ANGLE LIMIT
;
ALSTOP  LDA A     #OFFH
        STA A     LSBSAZ     ;STOPS AZ MOTOR
        STA A     LSBSEL     ;STOPS EL MOTOR
        RTS
;
;   "BCDSUB" SUBROUTINE
;           SUBTRACT 2 16-BIT BCD PACKED #"S
;           SUBTRACTS INDEXED ADDRESS FROM ACCA,ACCB
;           INDEX REG CONTAINS STARTING ADDRESS
;           ACCA,ACCB CONTAINS # TO BE SUBTRACTED FROM
;           RETURN RESULT IN ACCA,ACCB
;
;           **************************
;           *        9999            *
;           *       -IXRG            *
;           *       ------           *
;           *        DIFF            *
;           *       +   1            *
;           *       ------           *
;           *       DIFF+1           *
;           *       + BCD#           *
;           *       ------           *
;           *       ANSWER           *
;           **************************
;
;           BCS TSTS OV CONDITION
;           BCC TSTS NO OV CONDITION
;
BCDSUB  CLR       CARRY      ;RESET CARRY VALUE
        CMP A     0,X        ;IS CONTENTS OF ACCA BIGGER ?
        BHI       SUBT       ;BR IF MINUEND>SUBTRAHEND
        BNE       SWAP       ;BR IF MIN<>SUBTRAHEND
        CMP B     1,X        ;MSBYTE OF MIN=MSBYTE OF SUBTRAHEND
        BHI       SUBT       ;MIN>SUBTRAHEND
        BEQ       SUBT       ;MIN=SUBTRAHEND OK TO SUBTRACT AS IS
SWAP    PSH B     ;SAVE MIN TEMPORARILY
        PSH A
        DEC       CARRY      ;SET CARRY BYTE TO DENOTE OV
        LDA A     0,X        ;GET SUBTRAHEND
        LDA B     1,X
        TSX                  ;X REG POINTS TO MINUEND
SUBT    STA A     TEMPA      ;SUBTRACT SMALLER FR GREATER
        STA B     TEMPB      ;SAVE GREATER OF 2 #"S
        LDA A     #099H
        TAB                  ;ACCB=99
        SUB A     1,X        ;RESULT => ACCA,ACCB "9999"
        SUB B     0,X        ;SUBTRACT SMALLER NUM FROM 9999
        SEC
        ADC A     TEMPB      ;LS BYTE OF DIFF+LSBYTE OF GREATER NUM +1
        DAA
        PSH A                ;SAVE LS BYTE OF RESUL
        TBA                  ;MOVE MSBYTE OF DIFF INTO ACCA
        ADC A     TEMPA      ;MSBYTE OF DIFF+ MSBYTE OF > # + CARRY BIT
```

```
         DAA
         PUL B                 ;RETRIEVE LS BYTE OF RESULT
         TST        CARRY      ;ACCA,ACCB=RESULT OF BCD SUBTRACT
         BEQ        BACK       ;IF NO OV NO NEED CLEAN STACK
         INS                   ;CLEAN UP STACK
         INS
BACK     RTS
;
; "SHAENC" SUBROUTINE
;           READS SHAFT ANGLE ENCODERS
;
SHAENC   LDA A      MSBSAZ     ;READS AZIMUTH ANGLE
         LDA B      LSBSAZ
         STA A      MSBENC     ;STORES ANGLE IN TEMP LOC
         STA B      LSBENC
         ASL B                 ;SCALE DAC OUTPUT BY FACTOR OF 2
         ROL A
         STA A      DDRA2      ;OUTPUT MS 4 BITS OF AZ TO DAC
         STA B      DDRB2      ; "     LS " "     "  "  "  "
         LDA A      MSBENC     ;GET OLD A AND B
         LDA B      LSBENC
         LDX        #DIVISO
         JSR        DIVIDE     ;DIVIDES ANGLE BY 14,912
         LDX        #AZBCD
         JSR        BINBCD
         PSH A
         PSH B
;
;    ADDITION OF SHAENC SUBROUTINE - CK + AND - AZ LIMITS
;
         STA B      TEMPB      ;SAVES ACCB TEMPO
         LDA B      SIGN       ;START LIM CK
         CMP B      #2BH
         BEQ        PAL        ;BR IF AZ COORD +
         LDA B      TEMPB      ;AZ COORDS -
         LDX        #NAZLIM
         JSR        BCDSUB     ;ACCX-NAZLIM
         LDA A      CARRY
         BNE        SHA2       ;BR IF ACCX>NAZLIM
PAL1     LDA B      #0FFH      ;POSTITIONER EXCEEDED LIMIT
         STA B      LFLAGA     ;SET AZ LIMIT FLAG
         LDX        #MSG15
         JSR        ASCDIS     ;DISPLAY "ANGLE LIMIT EXCEEDED"
         JSR        ALSTOP     ;BR TO STOP BOTH MOTORS
         JMP        MSGB       ;WAIT 1 SEC AND GO CNTRL LP
PAL      LDA B      TEMPB      ;CK FOR + AZ LIMIT
         LDX        #PAZLIM    ;GET + AZ LIMIT
         JSR        BCDSUB     ;ACCX-PAZLIM
         LDA A      CARRY
         BEQ        PAL1
SHA2     LDA A      SIGN
         STA A      AZSIGN
         LDA A      #041H
         STA A      LETA
         LDA A      #05AH
         STA A      LETB
         PUL B
         PUL A
         LDX        #ANGLE
         JSR        BCDDIS     ;UPACK BCD ANGLE
         LDX        #DISAZ
         JSR        ASC2       ;DISPLAYS PACKED BCD ON PANEL
         LDA A      MSBSEL     ;READS EL ANGLE
         LDA B      LSBSEL
         STA A      MSBENC     ;STORE ANGLE TEMPORORILY
         STA B      LSBENC
```

```
ASL B
ROL A
STA A       DDRA3      ;OUTPUT MS 4 BITS OF EL TO DAC
STA B       DDRB3      ;OUTPUT LS 4 BITS OF EL TO DAC
LDA A       MSBENC     ;GET OLD A AND B
LDA B       LSBENC
LDX         #DIVISO
JSR         DIVIDE     ;DIVIDE ANGLE BY 14,912
LDX         #ELBCD
JSR         BINBCD     ;RET PACKED BCD #
PSH A
PSH B
STA B       TEMPB      ;HANDLES CHANGE IN COORD SYSTEM
LDA B       SIGN
```

DISTRIBUTION LIST

Dr. N. A. Kheir University of Alabama in Huntsville School of Engineering Huntsville, Alabama 35899	20
DRSMI-RDF	3
-R	1
-LP	1
-RPR	15
-RPT	1
US Army Material Systems Analysis Activity ATTN: DRXSY-MP Aberdeen Proving Ground, MD 21005	1

20

3
1
1
15
1

1

END

DATE
FILMED

10 — 83

DTIC